Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

High-Fidelity Image Synthesis from Lesion Maps using Diffusion Models

Author:
Xuan Zhao

Supervisor:
Dr. Benjamin Hou

Second Marker:
Prof. Ben Glocker

June 25, 2023

Abstract

Deep learning has been extensively used in the field of medical image synthesis and analysis. However, the full potential of deep learning is hindered by its huge demand for training data and high costs and difficulties to collect data with high-quality annotations. Past efforts have been focused on using GAN or autoencoders to generate synthetic samples for training the downstream models at larger scales. With the help of latest achievement made by Diffusion Models, recent work has demonstrated its ability to generate high-fidelity brain scans. This has shown the great potential of diffusion model in generating massive medical images.

Computed Tomography (CT) is currently the standard modality for detection and follow-up of pulmonary nodules, which need to be classified as suspicious or benign in order to aid early cancer detection. One challenge in this task is the lack of source images on which models of various tasks, such as segmentation or detection, need to be trained. In this work, we explore the use of Semantic Diffusion Models (SDM) to generate high-fidelity pulmonary CT images from segmentation maps. We utilize annotation information from the LUNA16 dataset to create paired CT images and masks, and assess the quality of the generated images using the Frechet Inception Distance (FID), as well as on two common clinical downstream tasks: nodule detection and nodule localization. Achieving improvements of 3.96% for detection accuracy and 8.50% for AP_{50} in nodule localization task, respectively, demonstrates the feasibility of the approach. Additionally, in this project, a Prior-net based segmentation, or prior-aware segmentation procedure, using diffusion model, has been experimented. Despite of the attempt being incomplete, relevant findings and results, as well as future work, have been documented.

Acknowledgements

I would like to thank my parents for their continuing support for me to attend higher education abroad over the years, as well as their encouragement to allow me pursue what I love to do

I would also like to thank my classmates at Imperial Department of Computing who have encouraged me during my time at Imperial and even given directly advises to me regarding this project. Without their day-to-day support, the completion of my thesis and the degree would not be possible.

More importantly, I would like to thank my partner, Peter Zhang, from University of Cambridge who generously provided multiple GPUs for me to train most of the models mentioned in this project, as well as assisted me to run downstream models with repetitions in parallel on multiple machines. I appreciate his understanding during the busiest time of my final year of study at Imperial.

Most importantly, I would like to thank my supervisor Benjamin Hou, without whom the completion as well as publication of this project would not be possible. Benjamin has been guiding me throughout the process, including giving out clear direction on learning the basics of diffusion model, offering helps in training the model due to demanding hardware requirement, working with me to publish the project as a short paper on Medical Imaging with Deep Learning (MIDL) 2023, and encouraging me to try different options and take notes even when the second task of this project is yet to produce expected results. His presence is the main source of motivation behind this project.

Contents

Chapter 1

Introduction

Lung cancer has been one of the leading causes of cancer deaths in the world for years [33]. Using Computed Tomography (CT), accurate detection and classification of pulmonary nodules is one of the main ways to diagnosing lung cancer. Before the emergence of deep learning in the past decades, diagnosis on lung nodules mainly relies on validations by clinical experts. Efforts of using deep learning to aid detection of malignant lung nodules have been made in the past. [28, 26, 39, 32] However, the available datasets are often not comprehensively labeled. Since nodules might be too small or insignificant, not all nodules in a CT scan slice are properly labeled. Some of the training dataset only contains few hundred samples, which might not be sufficient to train downstream models. [1] Image augmentation and synthesis are attempted in order to artificially increase sample size for training deep learning models. Past efforts include using image blending techniques [24], GAN [3], and auto-encoder [18] to generate new synthetic samples. However, GAN or auto-encoder have the issue of unstable training or inability to capture finer details [5, 9].

Recent development in diffusion models has shown promising result in its ability of synthesizing images with high fidelity and photo-realism. Within the field of medical imaging, diffusion model has been used to generate synthetic brain images [27] with finer details and higher resolutions. Diffusion model is also able to condition the image generation process by specifying few requirements, which provides high degree of control.

1.1 Objectives & Contributions

This project focuses on generating synthetic images of lung CT scans containing pulmonary nodules using diffusion models. The work is based on the publicly available dataset LIDC-IDRI which contains 1018 low-dose CT scans of the lung area and 7371 unique nodules marked by at least one radiologist. The generated images will be evaluated through downstream model metric improvements via dataset padding, as well as image quality metrics such as the FID score.

The report first introduces technical concepts of deep learning used in the past literature, then provides a thorough review of related work of generating synthetic medical images, and analyzes the techniques and performance implications of different methods. Next, the diffusion model and experiment process will be introduced. The result and findings will be included in the appendix section.

1.2 Ethical Considerations

All datasets used in this project are publicly available, with no risk of exposing individual privacy. The results and findings listed in this project will be public, thus providing ease to replicate the experiment and confirm its validity.

This project aims at generating synthetic lung CT scans, which might not fully capture the details of an authentic CT scan of the patients. In other words, the data distribution of the synthetic CT images might not generalize well compared to the ground-truth lung nodule data distribution. Care must be taken when using the generated data from this project. In principle, the nodules in the synthetic images must be checked with clinical experts in order to validate their correctness and malignancy.

Codes written in this project only use or reference open-source projects whose license will be fully reviewed and listed in the appendix. The repository will also include acknowledgements of authors of these code bases. The codes written in this project will also be publicly available.

1.3 Publication

The first task of this project has been published on Medical Imaging with Deep Learning (MIDL) 2023 conference as a short paper under the title *High-Fidelity Image Synthesis from Pulmonary Nodule Lesion Maps using Semantic Diffusion Model*. The author of this project will give an virtual poster presentation of relevant findings in this project.

Chapter 2

Background

2.1 Medical Concepts

2.1.1 Computed Tomography

Computed tomography (CT) is a common diagnostic imaging procedure using X-rays and computers to produce the images of the inner content of the body. It is mainly used to show the internal structures and tissues, such as; bones, muscles, blood vessels, and organs. A heated cathode releases high-energy electrons which in turn release their energy as X-ray radiation. The X-rays pass through the human body and hit a detector on the other side of the body. The detector will then absorbs X-rays and produce the image of the density of the scanned region.

CT scan contains more details and structural information than standard X-rays. The standard X-rays emits a beam of energy at the body part being studied, while a plate detector behind the body part records the residual radiation after the beam passes through skin, bones, and tissues. This provides information of a 2-dimensional distribution density of the region, but does not show the structure of the organs. In CT, the X-ray beams move in a circle around a body region, allowing views from different angle of the same organ or internal structure. The circular X-ray scans are sent to a computer which processes the data and displays either 2-dimensional or 3-dimensional images on the monitor. The CT scan image is then displayed in black and white pixels, with the intensity value being the tissue density measured in Hounsfield units (HU).

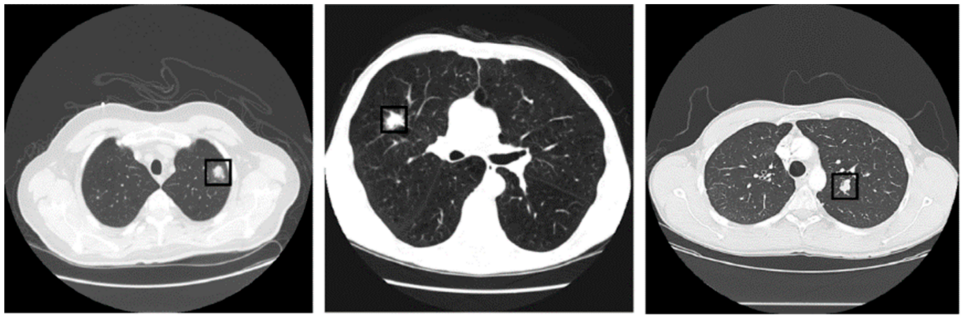

Figure 2.1: Sample CT scans from LIDC-IDRI dataset containing pulmonary nodules

2.1.2 Pulmonary Nodules

Pulmonary nodules are regions in the lungs that consists of cell clusters, or tissues, which are denser than normal lung tissue. Majority of the lung nodules are caused by scar tissues or some irritants in the air. However, it can still be an early sign of lung cancer, and its malignancy is usually determined by a clinic expert. Lung nodules can be divided into few categories: benign tumors, infections, inflammation, and malignant tumors. It also has characteristics such as size and shape. CT screening is one of the main ways to diagnose pulmonary modules with 40% likelihood being cancerous. Hence, it is vital to ensure the accuracy in the design of computer-aided pulmonary nodule diagnosis. In this project, the main focus is to generate both benign and malignant nodules CT images with various characteristics that resemble the ground truth in the LIDC-IDRI dataset.

The LIDC-IDRI dataset [1] establishes a publicly available reference for computer-aided pulmonary nodule diagnosis system. It contains 1018 chest CT scans from 1010 patients and within them are 7371 lesions marked as nodules by at least one radiologist. The nodules are marked by 4 radiologists and categorized into 3 groups: nodules $\geq 3mm$, nodules $\leq 3mm$, and non-nodules $\geq 3mm$. Out of the 7371 lesions, 2669 lesions were marked nodules $\geq 3mm$ by at least one radiologist, of which 928 received such marks from all four radiologists.

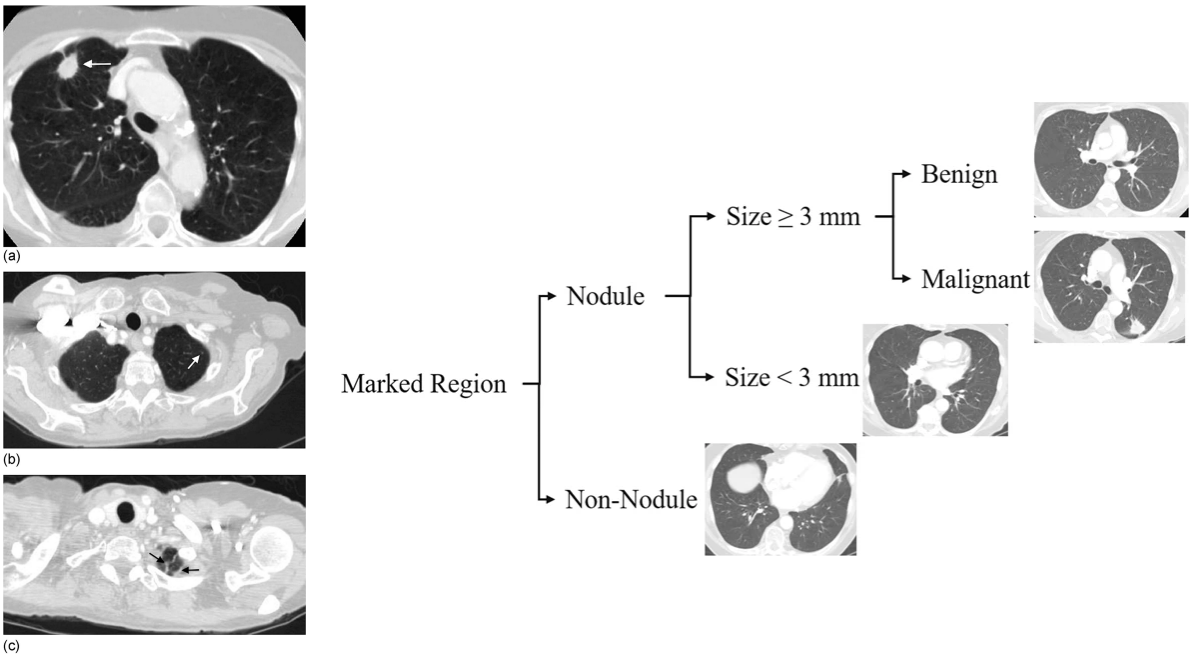

Figure 2.2: Left: Example of lesions considered to satisfy the LIDC/IDRI definition of a) nodule $\geq 3mm$, b) nodule $< 3mm$, and c) non-nodule $\geq 3mm$. Right: The categorization process of pulmonary nodules in LIDC/IDRI. Nodules larger than 3mm will then be further categorized into benign or malignant tumors.

2.2 Deep Learning Concepts

This section aims to introduce deep learning concepts that are used in past related work, as well as the diffusion model which is the focus of this project.

2.2.1 Autoencoders

An autoencoder is a neural network designed to learn the representation of unlabeled inputs by training itself to reconstruct the input data. It is an unsupervised learning technique consisting of an encoder and a decoder. It uses fully connected or convolution layers (the encoder E) as the hidden layer to learn the features of the input and another neural network (the decoder D) to produce an approximate reconstruction of the input. The reconstruction loss measures the differences between the reconstructed output and the original input, which is normally used in the training process for both decoder and encoder. Traditionally, autoencoders are used for dimensionality reduction or feature learning; recent breakthroughs combine the idea of latent variable models with autoencoders to form a powerful tool set of generative modeling.

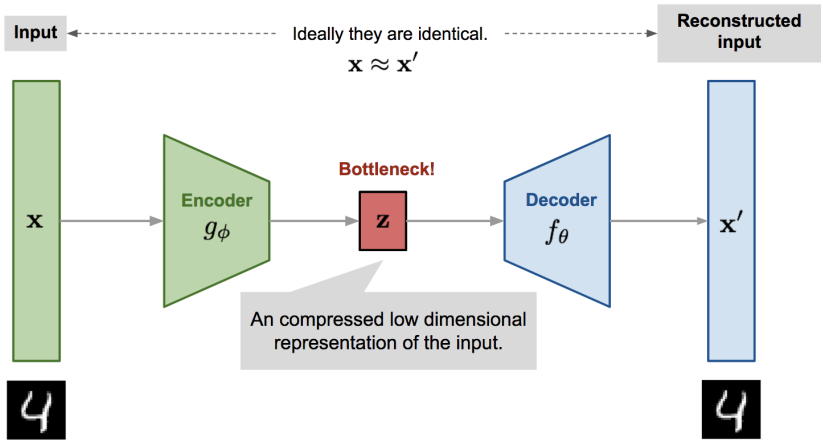

Figure 2.3: The typical architecture of an autoencoder. [38]

2.2.2 Variational Autoencoder

A variational autoencoder (VAE) [17] learns the approximate inference, namely a mean and a standard deviation for a Gaussian distribution, instead of the encoding of unlabeled inputs. The encoder of VAE will produce two vectors: one represents the means and the other represents the standard deviations of the distribution of each latent feature dimensions. Each pair of mean and standard deviation forms a Gaussian distribution in which sample values will be drawn. The sample values will then pass to the decoder network to generate the output, which is also pairs of mean and standard deviation. Finally, the reconstructed input will be sampled based on the output distributions. During the training process, the encoder is used to obtain the mean and standard deviation vectors, which are used to generate the latent samples, and the main objective is to maximize the variational lower bound associated with training data. During inference or generation process, the latent sample is drawn from the trained code distribution.

Despite the fact that VAE is elegant, simple to implement, and obtains state-of-the-art performance, samples generated from VAE tend to be blurry due to possible reasons such as the intrinsic effect of maximum likelihood which minimizes $D_{KL}(p_{data}||p_{model})$. [9]

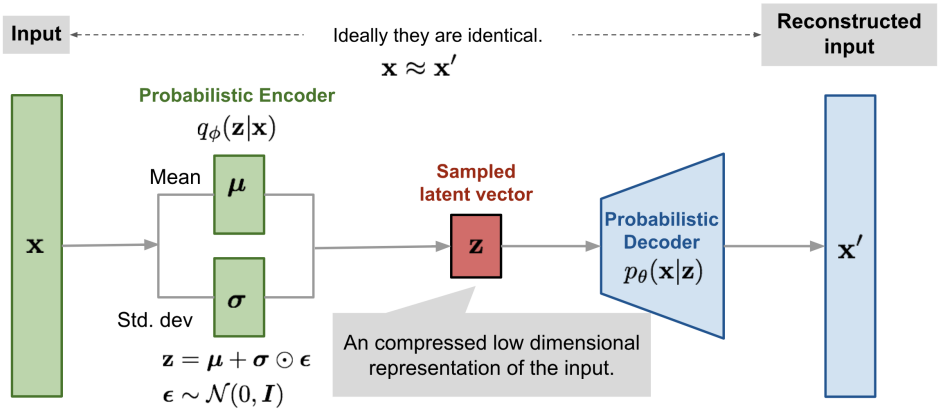

Figure 2.4: The typical architecture of a variational autoencoder. [38]

2.2.3 Generative Adverserial Network

A generative adversarial network is a class of machine learning framework where two neural networks, the generator and the discriminator, compete with each other in a minimax game in order to generate new data samples by the generator that are similar to the training dataset. The generator is responsible for generating new fake data samples, and the discriminator is responsible for differentiating between fake and real data samples. The goal of the generator is to fool the discriminator such that the discriminator cannot distinguish the fake samples from real samples, while the discriminator tries to find such distinction. The training process aims to achieve the objective function shown in Equation 2.1 while feeding samples from both the generated synthetic data made by the generator and the real data to the discriminator.

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

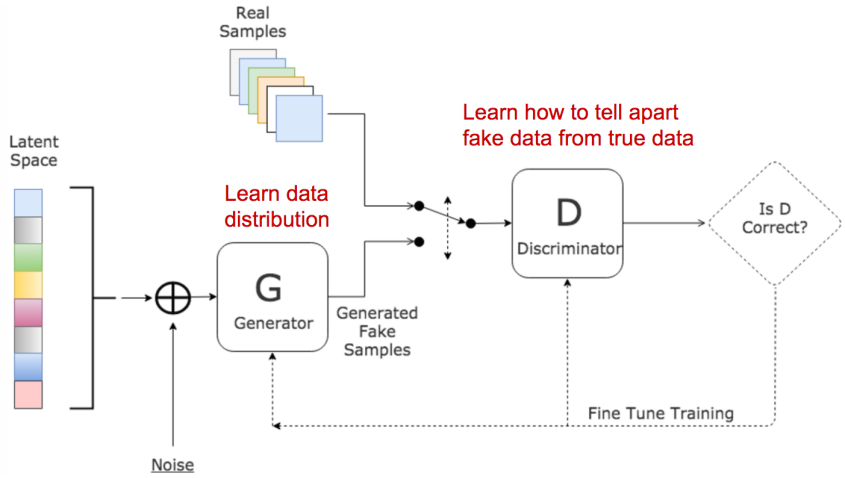

Figure 2.5: The typical architecture of a generative adverserial network. [37]

2.2.4 Diffusion Probabilistic Model

Diffusion model, or diffusion probabilistic model, is another class of generative models based on Markov chains that learns the latent structure of dataset by modeling the process in which data points diffuse through the latent space. Such process, known as the diffusion process, normally destroys the training data distribution, which helps the model to learn to recover the training data through reverse diffusion process. By reversing the process, the model can potentially generate new data.

Formally, diffusion models are latent variable models of the form $p_\theta(x_0) := \int p_\theta(x_{0:T}) d_{x_1:T}$ where $x_1, ..., x_T$ are latents of the same dimensionality as the data $x_0 \sim q(x_0)$. The joint distribution $p_\theta(x_{0:T})$ is called the reverse process and it is defined as a Markov chain with learned Gaussian transitions starting at $p(x_T) = N(x_T; 0, I)$:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) := N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.2}$$

The forward process incorporates an approximate posterior $q(x_{1:T}|x_0)$ that is fixed to a Markov chain representing the process of gradually adding Gaussian noise to the input data following a denoising schedule, which is a sequence of value $\beta_1, ..., \beta_T$.

$$q(x_{q:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) := N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I) \tag{2.3}$$

To train the diffusion model, we optimize the variational lower bound on the log-likelihood function:

$$\mathbb{E}[-\log p_\theta(x_0)] \leq \mathbb{E}_q[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}] = \mathbb{E}_q[-\log p(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}] := L \tag{2.4}$$

The loss function defined is similar to the loss used in the training process of a variational autoencoder (VAE). It can be re-written using the Kullback–Leibler divergence (KL divergence) form, and minimizing the KL divergence is equivalent to minimizing the loss function shown in Equation 2.4. [11] The definition of KL-divergence is given in Equation 2.6 and will be discussed in detail in Section 2.4.

2.2.5 Denoising Diffusion Probabilistic Models

In the field of computer vision, Ho et al. [11] has shown that denoising diffusion probabilistic models (DDPM) are capable of generating high-quality samples that are better than other types of generative models. When working on images, the diffusion process forms a Markov chain that gradually adds noise to the image until the signal is destroyed. The objective is to learn the transition of such Markov chain in order to reverse the diffusion process and thus generate new samples. While Ho et al. [11] has demonstrated its potential to produce high-quality images, it admits the fact that DDPM does not have a competitive log likelihoods compared to other likelihood-based models, such as VAEs. In terms of the diffusion schedule, the paper samples t uniformly for each image in each mini-batch during training without exploring potentially better schedule. Additionally, the forward diffusion process presented in Ho et al. [11] takes 1000 steps to perform,

which implies the slow training or inference speed. Iterating through the Markov chain is required for even producing one single sample, which takes around 20 hours to sample 50k images of size 32×32 on an Nvidia 2080 Ti GPU. However, it only takes GAN less than a minute on the same machine configuration. [34]

2.2.6 Improved DDPM

Nichol and Dhariwal [22] addresses aforementioned issues in DDPM and shows that with some improvements, DDPM can still achieve log-likelihoods competitive with other likelihood-based models and is able to scale on high-diversity datasets such as ImageNet. The paper introduces a new hybrid objective function which helps to learn the term $\Sigma_\theta(x_t, t)$. The variance term can then further improve the log-likelihood. In general, optimizing log-likelihood forces generative models to capture all of the modes of the data distribution [30] and small improvements in log-likelihood can result in drastic influences on synthetic sample quality and learned latent feature [10].

Another issue of DDPM is that it does not perform well on low resolution images below certain sizes. During the forward diffusion process, the image becomes excessively noisy at the last few steps. To address this issue, a better noise schedule can be used:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \tag{2.5}$$

This cosine schedule has a linear drop-off in the middle of the diffusion process and changes very little when the process is near $t = 0$ or $t = T$ to avoid abrupt changes in noise level. Compared to a linear diffusion schedule, cosine schedule preserves image latent information better and destroys the image signal more gradually.

The sampling speed of DDPM can also be improved significantly. Instead of using the same sequence of forward diffusion schedule time values used during training, it is also feasible to sample using an arbitrary sub-sequence of the diffusion schedule. During the training process, the value of $\Sigma_\theta(x_{S_t}, S_t)$ will be re-scaled for shorter diffusion process. In Figure 2.6, it shows that 100 sampling steps performs close to near-optimal FIDs of fully trained models.

2.2.7 Denoising Diffusion Implicit Models

In the original DDPM, the computation of the Markov chain of the forward and reverse diffusion process is slow. Denoising diffusion implicit models (DDIM), proposed in Song et al. [34], introduces non-Markovian diffusion processes while preserving the same training objective. This allows the reverse diffusion process to be sampled faster by skipping sampling steps since it is no longer necessary to traverse the entire chain of Markov process.

2.3 Related Work

This section evaluates past work which are highly relevant to the project. The discussion will follow the chronological order of date of publication of those past work.

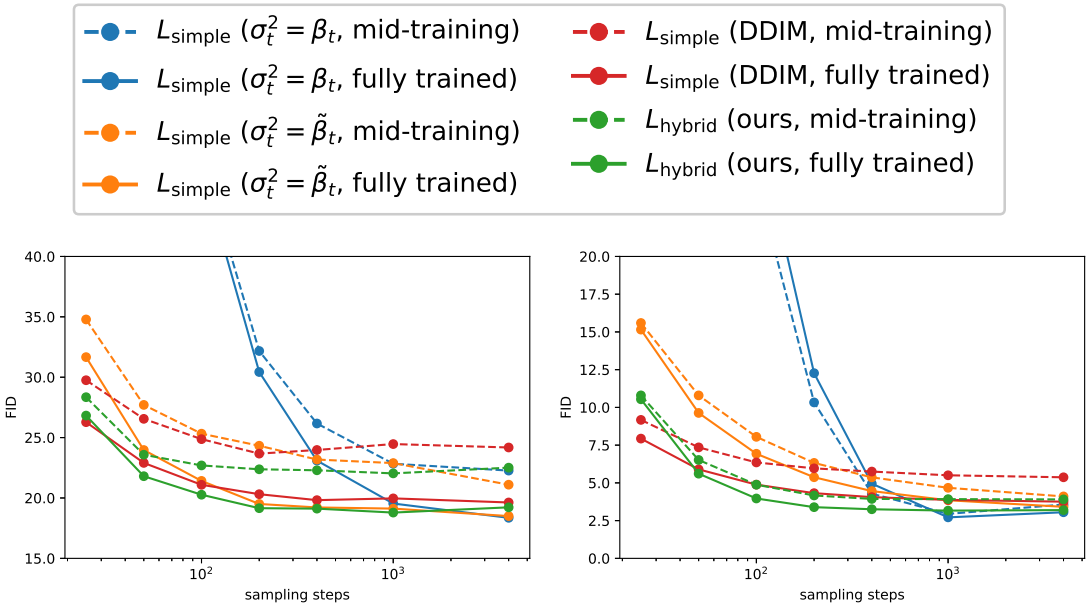

Figure 2.6: FID versus number of sampling steps, for models trained on ImageNet 64×64 (top) and CIFAR-10 (bottom). All models were trained with 4000 diffusion steps.

2.3.1 Gradient-based Methods

Before the massive application of deep learning in the medical imaging domain, synthetic pathology or features are generated using image blending or gradient field methods. Pezeshk et al. [24] introduces Poisson blending with gradient mixing in order to perform visually seamless insertion of pulmonary nodules in lung CT scans. The paper introduces a comprehensive new tool for insertion of lung nodules from one real CT scan into another in two steps: the user first needs to identify a region of interest by casually drawing a rectangular region around a nodule, and then selects the centre of the desired insertion area in the target slice. The tool will then use Poisson blending technique, which originally developed for computational photography, to patch the nodule on the real CT scan. Poisson blending is based on Poisson partial differential equation with Dirichlet boundary conditions. The technique of gradient mixing will further enhance the quality of patched nodule.

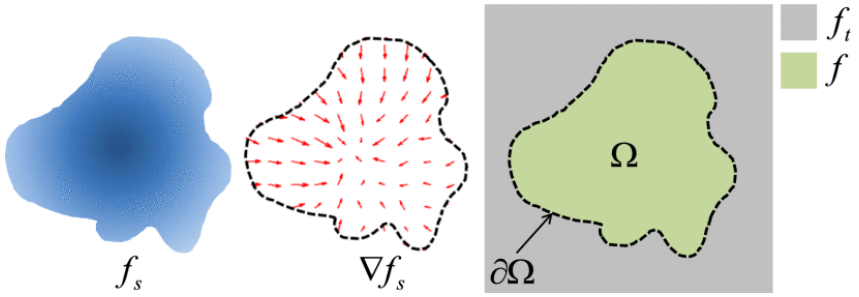

Figure 2.7: The Poisson blending method propoed in Pezeshk et al. [24]. f is the inward interpolation of f_t given the boundary conditions in $\partial\Omega$ and the guidance field ∇f_s derived from the source image f_s.

2.3.2 Methods using Autoencoder and GAN

More recent efforts focus on using deep learning methods to perform synthetic image generation, which can then apply to medical pathology image generation. Chuquicusma et al. [3] has used Deep Convolutional Generative Adversarial Networks (DC-GANs) to synthesize lung nodule CT scans that can easily deceive radiologists. The paper proposes 18 visual Turing tests to prove its validity. Later on, Kommrusch and Pouchet [18] uses autoencoders to generate 3D lung nodule images that match the features statistics of actual nodules as determined by an analysis program. The paper proposes the LuNG system, a synthetic lung nodule generator able to generate 3D shapes of lung nodules that fit within a broad learned category. The generator uses a neural network trained from a small set of seed images plus data augmentation. The bottleneck layer is a latent vector with a dimension of 3. Figure 2.8 shows the structure of the proposed autoencoder. After creating a well-trained auto-encoder, the network can be split into feature and generator network. The feature network can be used to map actual nodules into a latent feature space so that novel images similar to the actual input nodule can be created using the generator network.

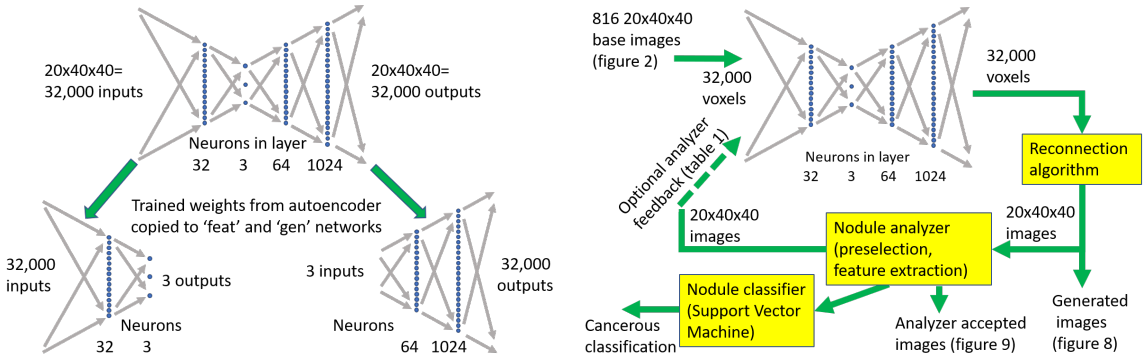

Figure 2.8: Left: the proposed autoencoder structure of the LuNG system. Right: interactions between trained autoencoder and nodule analyzer.

Later on, Park et al. [23] has demonstrated the ability of GAN to convert a semantic segmentation mask to a photo-realistic image. The paper proposed a new spatially-adaptive denormalization method (SPADE) which regulates the activation functions in the normalization layers. Previous methods of directly feeding the semantic map to the input layer and processing it through convolution, normalization and non-linear layers are sub-optimal since the information contained in semantic map tends to be washed away in normalization layer, leading to the network failing to capture the semantic information which generated image should incorporate.

2.3.3 Methods using Diffusion Model

The recent gain of popularity of diffusion model has motivated efforts of using it in semantic image synthesis as well as medical image generation. Wang et al. [36] has used DDPM to perform semantic image synthesis, which produced photo-realistic images of much better quality compared to images generated through GAN or VAE methods. [5] The paper presents a novel framework, the Semantic Diffusion Model (SDM), which transforms the sampled Gaussian noise into a realistic image through the iterative denoising, or reverse

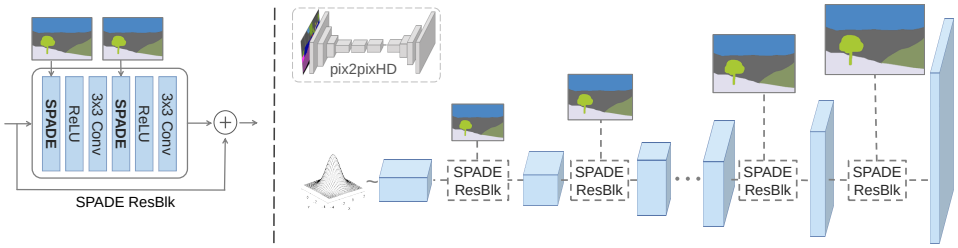

Figure 2.9: The architecture of SPADE method proposed by Park et al. [23].

diffusion process. The generation process is a parameterized Markov chain, and each step involves noise estimation from the noisy image in the last step also conditioned on the semantic label map. According to the estimated noise, a less noisy image with semantic-related content is generated by the posterior probability formulation in reverse diffusion. Previously, the conditional DDPMs directly concatenate the condition information, or the semantic label map, with the noisy image as input of the denoising network. This approach does not fully leverage the information in the input semantic mask, which leads to generated images in low quality and semantic relevance. [23] In this paper, the noisy image is fed into the encoder of the denoising network while the semantic layout is embedded into the decoder of the denoising network by multi-layer spatially-adaptive normalization operators. This highly improves the quality and semantic correlation generated images.

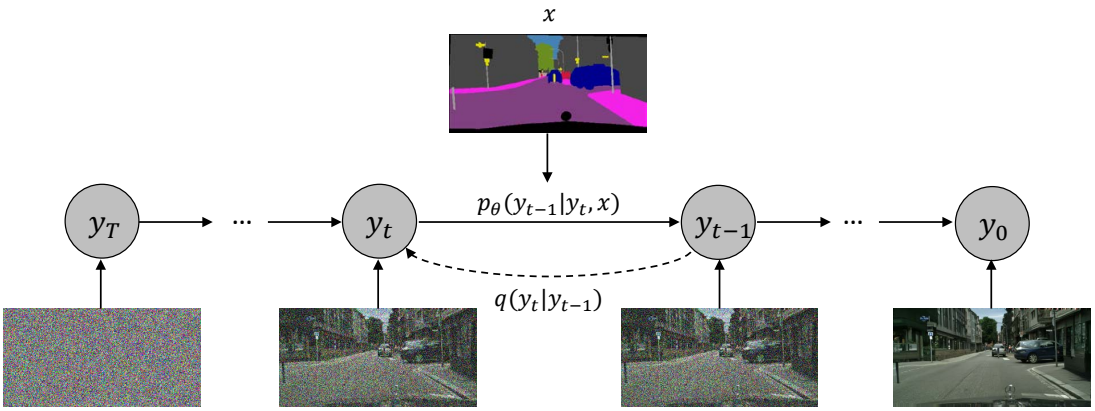

Figure 2.10: The diffusion process of SDM with semantic information used in leveraging the diffusion process.

Another highly relevant latest effort is using diffusion models to generate brain imaging. Pinaya et al. [27] creates synthetic MRI images of the adult human brain using the latent diffusion models. The paper uses about 30,000 training images from the UK Biobank to train the models. The image generation is also conditioned on age, gender, ventricular volume, and brain volume relative to the intracranial volume in order to generate realistic examples of brain scans. Latent diffusion model [31] is the combination of using the autoencoders to compress input data into lower dimensional latent representation and the the generative modeling aspect of the diffusion models. The encoder helps to compress the brain images to latent feature map of size $20 \times 28 \times 20$. After compression, the latent feature map is used as input to the diffusion model. The sampling quality of the experiments in the paper is shown in Figure 2.12. The generated brain images are also available publicly as the LDM 100k Dataset. [25]

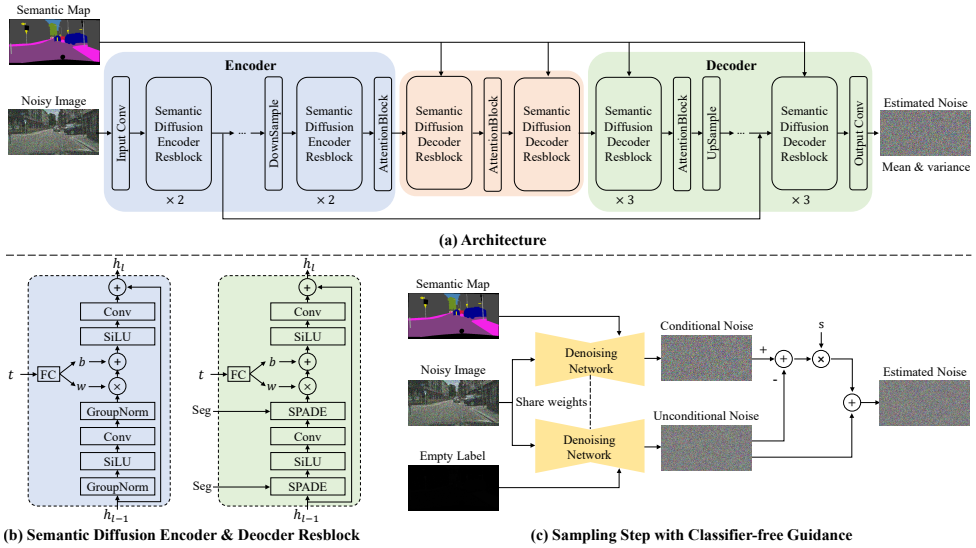

Figure 2.11: The overall architecture of SDM.

2.4 Evaluation Metrics

This section will talk about some common evaluation metrics used in the general literature of synthetic image analysis, as well as the classification and localization task evaluation metrics used in this project.

2.4.1 Inception Score

Many generative models, such as GAN, do not have a comparable objective function, which makes it hard or even impossible to compare the performance across different models. A naive way to evaluate the generated images is to use a human annotator to distinguish between generated images and real images. However, this evaluation metric is potentially inaccurate, as the differences in responses before and after giving the annotators about clues of fake images are drastic. As a better and automatic way to evaluate the quality of generated images, inception score method is proposed. An Inception model is applied to every generated image to get the conditional label distribution $p(y|x)$ with low entropy. [35] The model is also expected to generate large variety of images, hence the marginal $\int p(y|x = G(z))dz$ should have high entropy. The definition of inception score is thus the combination of those two terms using Kullback–Leibler divergence (KL-divergence): $\exp(\mathbb{E}_x \mathrm{KL}(p(y|x)||p(y)))$. The definition of KL-divergence is shwon in Equation 2.6. This metric has been found to correlate well with human judgments, provided that this metric is calculated on a large enough number of samples since the second term needs to evaluate the diversity of the dataset.

$$D_{KL}(P||Q) = \sum_{x \in \mathbb{X}} P(x) \log(\frac{P(x)}{Q(x)}) \tag{2.6}$$

2.4.2 Fréchet Inception Distance

While the inception score only evaluates the distribution of generated images, the Fréchet Inception Distance (FID) metric compares the distribution of generated images with the

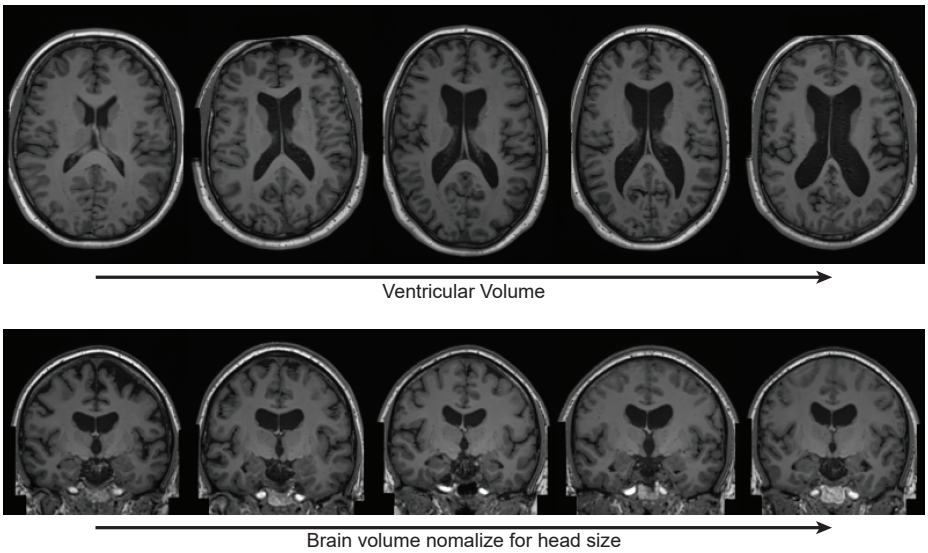

Figure 2.12: Conditioned sampling varying the ventricular volume and the brain volume normalized by the intracranial volume. In both rows, we kept the other variables constant.

distribution of a set of ground-truth images. For any two probability distrivubtions μ, v over $\mathbb{R}^n$ having finite mean and variances, their Fréchet distance is:

$$d_F(\mu, v) := (\inf_{\gamma \in \Gamma(\mu,v)} \int_{\mathbb{R}^n \times \mathbb{R}^n} ||x - y||^2 d\gamma(x, y))^{\frac{1}{2}} \tag{2.7}$$

where $\Gamma(\mu, v)$ is the set of all measures on $\mathbb{R}^n \times \mathbb{R}^n$ with marginals μ and v on the first and second factors respectively. For a two multi-dimensional Gaussian distributions $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu', \Sigma')$, their Fréchet distance is explicitly solvable as:

$$d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma'))^2 = ||\mu - \mu'||_2^2 + \mathrm{tr}(\Sigma + \Sigma' - 2(\Sigma^{\frac{1}{2}} \cdot \Sigma' \cdot \Sigma^{\frac{1}{2}})^{\frac{1}{2}}) \tag{2.8}$$

The Fréchet Inception Distance is then calculated as follows: given a function $f : \Omega_X \to \mathbb{R}^n$ and two datasets $S, S' \subset \Omega_X$, compute $f(S), f(S') \subset \mathbb{R}^n$ and fit two Gaussian distributions $\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma')$ for $f(S)$ and $f(S')$, respectively, and finally calculate the FID score according to Equation 2.8. The choice of f can be customized, but the most practical choice is an Inception v3 model [35] without its final classification layer trained on the ImageNet dataset.

The FID metric came out in 2017 and has since become the standard metric to evaluate the quality of generative models, such as the high-resolution StyleGAN1 and StyleGAN2 [15, 16].

2.4.3 Accuracy, F-score, Average Precision and Recall

Commonly for classification tasks, accuracy and F1 score are calculated and used as indication of the performance of machine learning models.

Accuracy measures the overall correctness of a classifier by calculating the ratio of correctly classified instances to the total number of instances. It is represented as a percentage. While accuracy is a commonly used metric, it can be misleading in scenarios where the classes are imbalanced. For example, if 95% of the data belongs to class A and

only 5% to class B, a classifier that always predicts class A will achieve 95% accuracy even if it fails to predict any instances of class B.

Precision quantifies the ability of a classifier to correctly identify positive instances from the total predicted positive instances. It is the ratio of true positives (correctly predicted positive instances) to the sum of true positives and false positives (instances wrongly predicted as positive). Precision focuses on minimizing false positives and is particularly useful when the cost of false positives is high. A higher precision value indicates fewer false positives.

Recall (Sensitivity or True Positive Rate) measures the ability of a classifier to correctly identify positive instances from the total actual positive instances. It is the ratio of true positives to the sum of true positives and false negatives (instances wrongly predicted as negative). Recall is useful when the cost of false negatives is high, as it emphasizes minimizing those errors. A higher recall value indicates fewer false negatives.

Specificity (True Negative Rate) measures the ability of a classifier to correctly identify negative instances from the total actual negative instances. It is the ratio of true negatives (correctly predicted negative instances) to the sum of true negatives and false positives. Specificity is the complement of the false positive rate and is useful when the cost of false positives is high. A higher specificity value indicates fewer false positives.

The F1 score combines precision and recall into a single metric to provide a balanced evaluation of a classifier. It is the harmonic mean of precision and recall and is calculated using Equation 2.13. The F1 score ranges from 0 to 1, where 1 represents perfect precision and recall. It is commonly used to strike a balance between precision and recall.

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.9}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.11}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.12}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.13}$$

In the context of evaluating an object localization task, average precision and average recall are commonly used metrics to assess the performance of a model in accurately localizing objects within an image. These metrics take into account both the precision and recall values at different levels of overlap between the predicted bounding boxes and the ground truth bounding boxes.

Average Precision (AP) measures the average precision values at different levels of overlap (IoU - Intersection over Union) between the predicted bounding boxes and the ground truth bounding boxes. IoU is calculated by dividing the area of intersection between the two bounding boxes by the area of their union. The precision-recall curve is plotted by sorting the predicted bounding boxes based on their confidence scores and calculating precision and recall values at different IoU thresholds. The AP is then calculated by taking the average precision values over all the recall levels. A higher AP indicates better object localization performance.

Average Recall (AR) measures the average recall values at different IoU thresholds. Similar to AP, the precision-recall curve is plotted, but this time the recall values are

considered as the x-axis, and precision is plotted against each recall level. AR is calculated by taking the average recall values over all the IoU thresholds. A higher AR indicates better object localization performance.

It's worth noting that there are different variants of AP and AR, such as AP at different IoU thresholds (e.g., AP_{50}, AP_{75}). These variations provide additional insights into the model's performance at different levels of localization accuracy and detection thresholds. Figure 2.13 illustrates the idea of IoU and the method to calculate AP/AR at a selected IoU level.

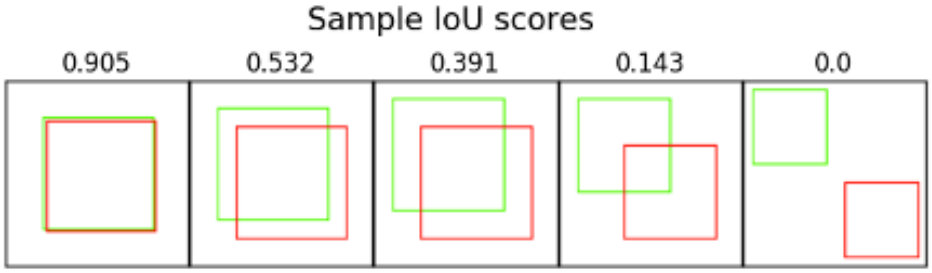

Figure 2.13: Illustration of different levels of Intersection-over-Union (IoU). Once a level of IoU is selected, e.g. 0.5, a bounding box prediction that overlaps more than 50% of the ground truth bounding box will be treated as a positive prediction and thus the corresponding AP and AR can be calculated.

2.4.4 Wilcoxon Rank-sum Test

The Wilcoxon rank-sum test, also known as the Mann-Whitney U test, is a non-parametric statistical hypothesis test used to determine if two independent samples or data come from populations with the same distribution. It is an alternative to the t-test when the assumptions for parametric tests, such as normality or equal variances, are not met. It works by assigning ranks to the combined data from the two samples, regardless of their group membership. The ranks are assigned based on the combined order of the observations, from lowest to highest. Then, the sum of the ranks for one of the samples is calculated. The test statistic, U, represents the smaller of the two possible sums of ranks and is used to assess the null hypothesis. The null hypothesis of the test states that the distributions of the two samples are equal, while the alternative hypothesis states that there is a difference between the distributions. The test determines whether the observed difference in ranks is statistically significant or could have occurred by chance.

The Wilcoxon rank-sum test has several variations depending on the specific scenario and assumptions. For example, the one-sided version of the test can be used if there is an a priori expectation that one sample will have consistently higher ranks than the other. The test can also be applied to matched pairs or dependent samples by using the Wilcoxon signed-rank test.

The test provides a p-value as the outcome, which indicates the probability of observing the given rank difference or a more extreme difference if the null hypothesis is true. If the p-value is below a pre-determined significance level (often 0.05), the null hypothesis is rejected, suggesting evidence of a significant difference between the two sample data

distributions. It is particularly useful when analyzing ordinal or skewed data, small sample sizes, or when the assumptions of parametric tests are not satisfied.

In this project, the Wilcoxon rank-sum test is used to determine whether the synthetic images generated by the diffusion model have indeed improved the downstream model. Specifically, the Wilcoxon rank-sum test is used to determine whether the accuracy of running the downstream nodule detection, localization, and classification tasks multiple times, both before and after padding the dataset, has indeed changed.

Chapter 3

High-Fidelity Pulmonary Nodule Image Synthesis via Diffusion Model

3.1 Problem Statement

As mentioned previously, developing robust and accurate computer-assisted diagnosis models using deep learning is vital in order to perform early diagnosis of lung cancers, but such models often requires large-scale and diverse medical datasets with high-quality annotations. Generations using autoencoder or GauGAN have been experimented in the past, but Dhariwal and Nichol [5] shows that diffusion model generates synthetic images of better quality and fidelity compared to GAN. For this part of the project, we use the Semantic Diffusion Model guided by the segmentation mask of the lung scan to generate synthetic samples containing pulmonary nodules, and evaluate the quality of synthetic samples via two downstream tasks.

3.2 Dataset padding and downstream model improvement

3.2.1 Implementation via SDM

The Semantic Diffusion Model (SDM) [36] involves a diffusion process guided by the segmentation mask of the target image we want to syntheize. As discussed in Section 2.3.3, SDM is different from previous conditional diffusion models that directly feed in the segmentation mask and the noisy image as input to a U-net; instead, it feeds only the noisy image to the encoder of the U-net while feeding the segmentation mask later in the decoder by using the multi-layer spatially-adaptive normalization operators.

One of the biggest advantage of SDM is the conditioning of segmentation masks during the diffusion process, which gives significant control over the shapes and sizes of lung nodules that we might want to generate.

In our experiment, we first generate the segmentation masks by finding the index of the pathological slices in the LUNA16 dataset containing the centroids of the marked nodules. Next, we create the masking of the pathological slices. To create nodule masks, the nodules are first cropped spherically based on the centroid and diameter information from the provided annotations. A manual OTSU threshold is then applied to each cropped region-of-interest to get the final masks. The intensities of the cropped pixels are clustered

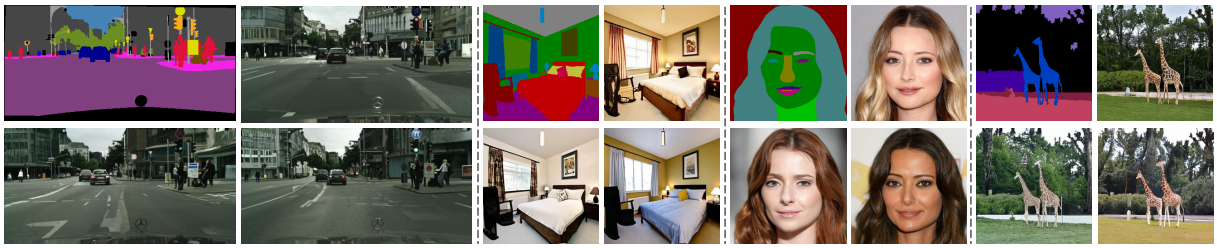

Figure 3.1: Examples of quality and diversity of images generated by SDM over 4 different datasets. Left-top is the input mask, and the rest of the three images are examples of generation via SDM.

using the K-Means algorithm with two centers, and the threshold is selected as the average of these centers. Additionally, a 'body mask' is also generated, which comprises the entire patient's body. Each slice is intensity thresholded at 127, followed by a morphological hole fill process. The largest connected region is then selected. The final mask is then composed of the structures in this order; background, left lung, right lung, trachea, body mask, and nodule if one is present. Listing 1 shows the Python pseudocode corresponding to the mask generation process. During this process, the CT window is set between [-1,000, 400] to accentuate the lung features, and operations are performed in 2D. Slices are considered only if they contain lung structure as nodules do not exist outside these regions. We also save healthy images by sampling every 4 slices in the CT volume that are nodule free.

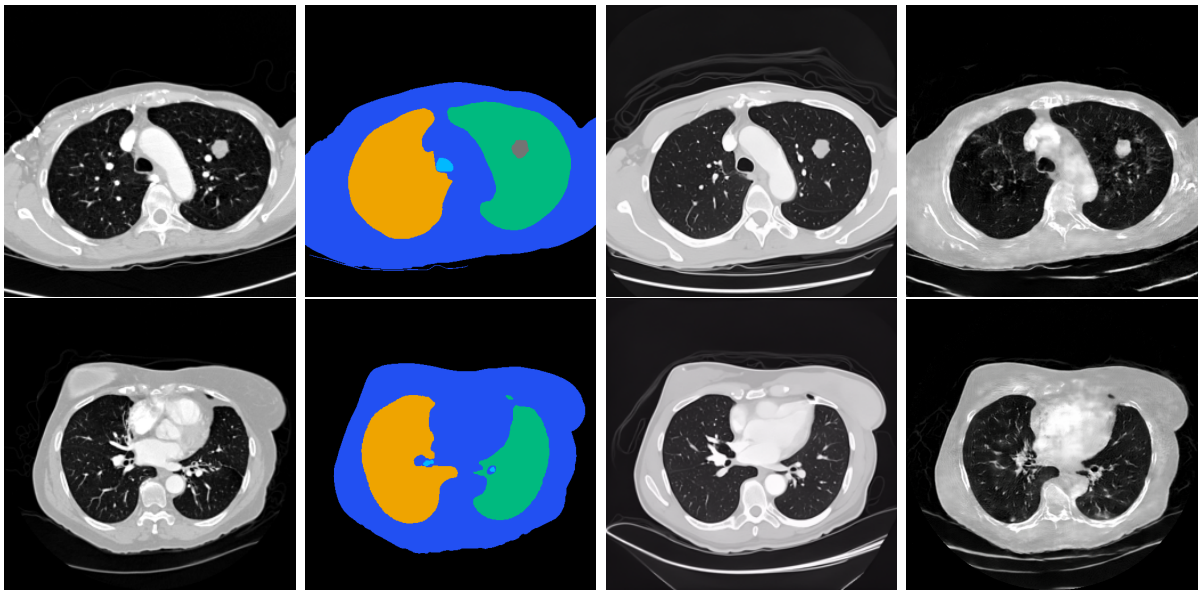

Figure 3.2: Example images generated by SDM and SPADE. L-to-R: CT image, CT segmentation mask, SDM image, SPADE image. Top: Nodule slice. Bottom: Nodule-free slice. In the CT segmentation mask: black - background, orange - left lung, green - right lung, cyan - trachea, blue - body mask.

The data pre-processing method above results in 1,139 slices containing nodules, and 32,040 healthy slices without nodules. We then use those slices to train an SDM which learns the distribution and characteristics of the five masked regions. The SDM is trained

```python
def make_mask(image, center, diam):
    mask = np.zeros_like(image, dtype=np.uint8)
    mask = cv2.circle(mask,
    ↪  (abs(int(center[0])),abs(int(center[1]))),int(abs(diam//2)), 255, -1)
    return mask

def make_body_mask(image):
    _, body_mask = cv2.threshold(image,127,255,cv2.THRESH_BINARY)
    body_mask = ndimage.binary_fill_holes(body_mask).astype(int)

    labels_mask = measure.label(body_mask)
    regions = measure.regionprops(labels_mask)
    regions.sort(key=lambda x: x.area, reverse=True)
    if len(regions) > 1:
        for rg in regions[1:]:
            labels_mask[rg.coords[:,0], rg.coords[:,1]] = 0
    labels_mask[labels_mask!=0] = 1
    return labels_mask

# node_x, node_y, node_z, diam are given
# origin and space are medical image metadata
center = np.array([node_x, node_y, node_z])
# nodule center and diameter in voxel space (x,y,z ordering)
v_center = np.rint((center - origin)/space).astype('int')
v_diam = np.ceil((diam / space)).astype('int')

idx = int(v_center[2])  # slice idx centered on nodule

# masking 3, 4, 5 corresponds to left lung, right lung, trachea
image_slice = image_arr[idx]
l_lung_mask = (mask_arr[idx]==3).astype('uint8')
r_lung_mask = (mask_arr[idx]==4).astype('uint8')
trachea_mask = (mask_arr[idx]==5).astype('uint8')

# Creating the masks of the body and the nodule
body_mask = make_body_mask(image_slice)
nodule_mask = make_mask(image_slice, v_center, v_diam[0])

# Using binary thresholding to extract the nodule mask
nodule_mask = cv2.bitwise_and(image_slice, image_slice,
↪  mask=cv2.dilate(nodule_mask, kernel=np.ones((5,5))))
pts = nodule_mask[nodule_mask>0]
kmeans2 = KMeans(n_clusters=2).fit(np.reshape(pts,(len(pts),1)))
centroids2 = sorted(kmeans2.cluster_centers_.flatten())
threshold2 = np.mean(centroids2)
_, nodule_mask = cv2.threshold(nodule_mask, threshold2, 1, cv2.THRESH_BINARY)

# Stacking aforementioned masks together in the following order
total_mask = np.stack([np.zeros((512,512)), body_mask, l_lung_mask*2,
↪  r_lung_mask*3, trachea_mask*4, nodule_mask*5], axis=-1)
total_mask = np.argmax(total_mask, axis=-1)
```

Listing 1: Pseudocode for the segmentation mask generation process, preparing for using the SDM. total_mask is the final segmentation mask. The final mask is stored as a png image whose visual appearance is all black since all pixel values are only within range 0 to 5 out of 255.

```
#!/bin/bash
python image_train.py --dataset_mode ade20k --lr 1e-4 --batch_size 4
  --attention_resolutions 32,16,8 --diffusion_steps 1000 --image_size 256
  --learn_sigma True --noise_schedule linear --num_channels 256
  --num_head_channels 64 --num_res_blocks 2 --resblock_updown True
  --use_fp16 True --use_scale_shift_norm True --use_checkpoint True
  --num_classes 6 --class_cond True --no_instance True
```

Listing 2: The actual training configuration used to train the SDM.

for 80,000 rounds with 1,000 diffusion steps for each round using a linear schedule, with learning rate being 1e-4 and batch size being 4 due to GPU memory limitation. The image size is set to 256 x 256 to speed up the DDPM generation process, and the number of classes involved is set to 6 since there are 6 types of segmentation masks, including the background. The rest of the training configuration stays the same as the default training configuration for the Ade20k dataset in the SDM repository, as shown in Listing 2.

3.2.2 Implementation with Conditional Masking on Malignancy Score

In addition to the locations and sizes of the pulmonary nodules, the original LIDC dataset [1], of which LUNA16 is a subset, also contains the malignancy score for each nodule. The score ranges from 0 to 4, with 0 being benign and 4 being the most malignant. With the consideration of malignancy score, 822 scans in total are utilized while other slices possibly miss annotations of valid malignancy scores from radiologists. Based on Gao et al. [7] which proposes a sound method for distinguishing benign and malignant pulmonary nodules, we have chosen 3.0 to be the threshold to separate benign and malignant nodules. This results in 435 scans with benign nodules and 387 scans with malignant nodules. Following the data pre-processing steps in Section 3.2.1, except that the nodule mask is now divided into benign nodule mask (pixels assigned as value 5) and malignant nodule mask (pixels assigned as value 6), the new malignancy-aware segmentation masks are generated. The SDM is then trained as before, but conditioned on the malignancy of the lung nodules. By doing this, the SDM will learn the shape and size differences between the benign and malignant nodules, and thus will be better at distinguishing between those two. Listing 3 shows the relevant modifications to the code in Listing 1 that incorporate malignancy class information into the masking. Figure 3.3 shows example images of different malignancy scores ranging from 1.0 to 5.0.

During the process of generating conditional masks, a `average_malignancy.csv` file has been generated and used as the reference to obtain the malignancy score of each nodule. The `average_malignancy.csv` file is generated using the LIDC dataset which is the superset of LUNA16. The LIDC dataset contains the comprehensive information of each annotation made by different radiologists on each lung scan, which are stored in XML format. A.3 shows the script that generates the `average_malignancy.csv`. The malignancy score needs to be averaged since each nodule might be annotated by multiple radiologists.

```
...
# masking 3, 4, 5 corresponds to left lung, right lung, trachea
image_slice = image_arr[idx]
l_lung_mask = (mask_arr[idx]==3).astype('uint8')
r_lung_mask = (mask_arr[idx]==4).astype('uint8')
trachea_mask = (mask_arr[idx]==5).astype('uint8')

# Creating the masks of the body and the nodule
body_mask = make_body_mask(image_slice)
nodule_mask = make_mask(image_slice, v_center, v_diam[0])

# Using binary thresholding to extract the nodule mask
...

# Added/modified lines below to incorporate benign and malignant class into the
↪  mask
malignancy = -1
with open("average_malignancy.csv", "r") as f:
    csv = pd.read_csv(f)
    z_coord = float(row['coordZ'])
    temp = csv[csv['seriesuid'] == seriesuid]
    malignancy = temp[temp['z_coord'] == z_coord]['avg_malignancy']
    malignancy = malignancy.values[0] if len(malignancy) > 0 else -1
    f.close()

if malignancy == -1:
    continue

benign_mask = 5 if malignancy < 3 else 0
malignant_mask = 6 if malignancy >= 3 else 0
total_mask = np.stack([np.zeros((512,512)), body_mask, l_lung_mask*2,
↪  r_lung_mask*3, trachea_mask*4, nodule_mask * benign_mask, nodule_mask *
↪  malignant_mask], axis=-1)
total_mask = np.argmax(total_mask, axis=-1)
```

Listing 3: Modified version of the code in Listing 1. total_mask is the final segmentation mask. average_malignancy.csv is a file prepared by the script in Appendix A.3. The final mask is stored as a png image whose visual appearance is all black since all pixel values are only within range 0 to 6 out of 255.

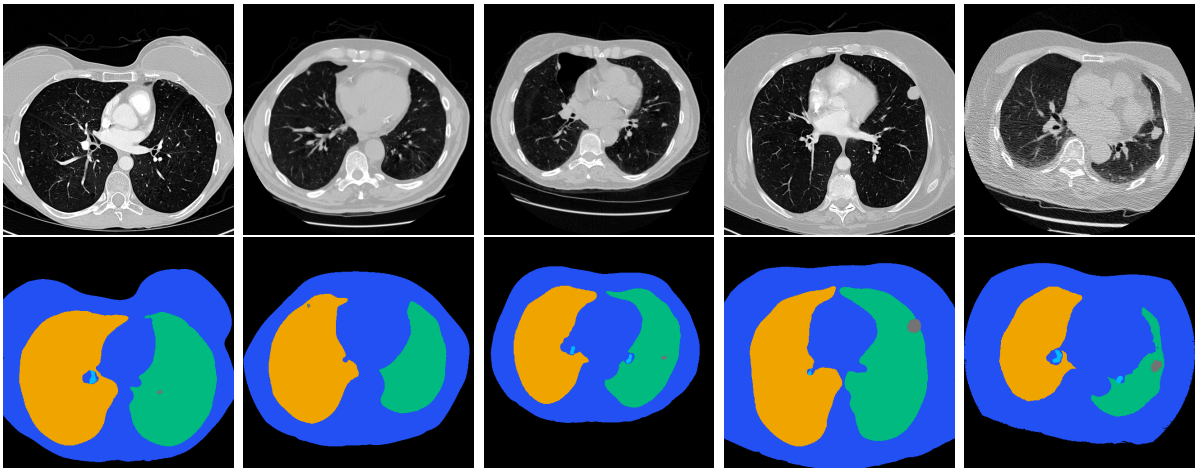

Figure 3.3: Top: lung scan containing nodules. Bottom: corresponding segmentation mask. From left to right: malignancy score from 1.0 to 5.0. The size of the nodule increases along with the score.

3.2.3 Improved Downstream models/tasks

Two common downstream medical image processing tasks have been chosen to evaluate the quality and validity of the synthetic images: nodule detection and localization. For both tasks, MMPreTrain [4] framework has been used.

The nodule detection task is to determine whether a cropped 32 x 32 patch is nodule or non-nodule. Essentially, this is a classification task. For this downstream task, we cropped out the nodules such that the centroid of the nodule locates in the center of the 32 x 32 region. For the baseline model, we use all 1,139 nodule crops and randomly select 1,139 non-nodule crops to train the detection network. Note that by definition, a non-nodule part can also potentially be a small cluster of tissue, which might look like a nodule. (See Figure 2.2) A Squeeze-and-Excitation ResNet (SE-ResNet) architecture [12] is chosen as the detection task network architecture. Figure 3.4 shows the difference between a normal residual block and a SE-Residual block. The SE-ResNet is pre-trained and the configuration `seresnet101_8xb32_in1k` is selected, whose detail is shown in Table 3.1. The SE-ResNet is trained for 50 epochs with 128 iterations per epoch. The learning rate is set initially to 0.1 and decays using discrete staircase per 30 epochs over time, and the batch size (samples learned per iteration) is set to 8.

The nodule localization task involves detecting the bounding boxes of nodules on a 2D slice, which is implemented by a Faster R-CNN model [12] by choice. The ground truth bounding boxes are first generated according to the diameters and centroid locations of the nodules (which are by definition spheres): the boxes are centered at the centroid locations of each nodule, and the side length of the box is set to the diameter of that nodule. Essentially, the bounding box is created such that the nodule is the inscribed circle of the box. Notice that the actual nodule segmentation is generated by the pre-processing step mentioned in Section 3.2.1; the nodule definition of centroid locations and diameters here is from the provided LUNA16 dataset. The Faster R-CNN, whose overall architecture is shown in Figure 3.4, is trained over 12 epochs with 510 iterations per epoch. The learning rate follows a linear decay schedule that adjusts over time during training. The configuration `faster_rcnn_x101_32x4d_fpn_1x_coco` is chosen, and Table 3.1 shows the detail of this configuration.

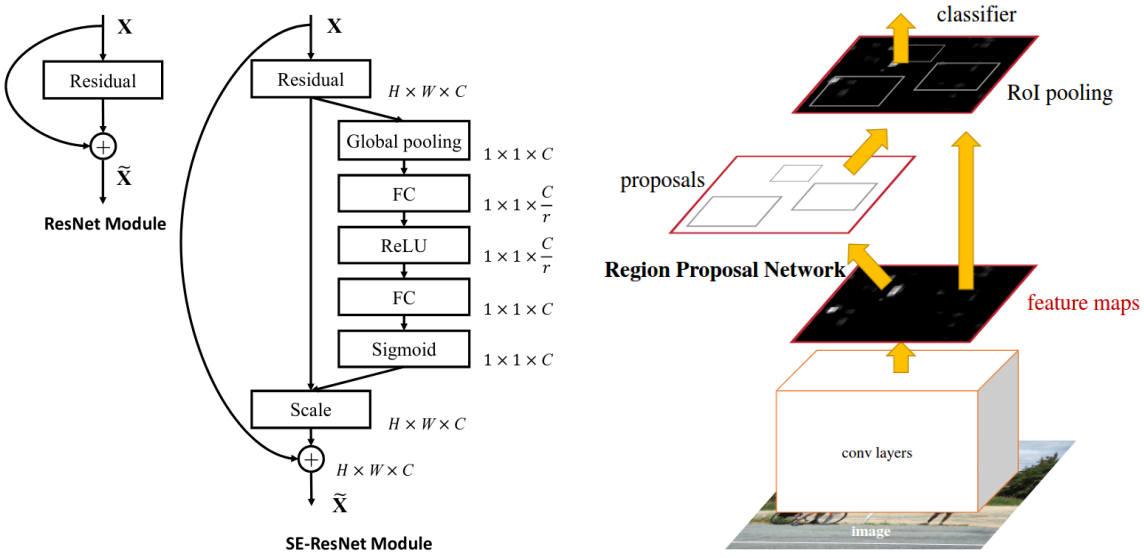

Figure 3.4: Left: comparison between a normal Residual Block and the Squeeze-and-Excitation Residual Block. Right: the overall architecture of Faster R-CNN.

Details of the training and inference configuration can be found in the Appendix A.2.

	Mem (GB)	Pre-trained model Perf.	Learning rate Schedule	Epoch	Batch Size
I	5.8	Top-1: 78.26%, Top-5: 94.07%	Discrete Staircase decay by factor of 0.1 every 30 epochs	50	8
II	7.2	Box AP: 41.2%	Linear Decay by factor of 0.1 at epoch 8 and 11, with warm-up ratio 0.001 in the first 500 iters.	12	2

Table 3.1: Details of the configurations of the nodule classification and localization tasks. I: nodule detection/classification task, II: nodule localization task.

Additionally for the SDM masks conditioned on malignancy score, a nodule classification model is chosen and evaluated as well. Different from the first task of nodule detection where the goal is to distinguish between nodule and non-nodule patches, the third task is to classify a nodule as benign or malignant, whose ground truth is the average malignancy score provided in the `average_malignancy.csv` file. The model configuration used for the third task is the same as the first task, differing only in that the third task classifies nodule into 3 classes (non-nodule, benign, and malignant), whereas the first task classifies nodule into only 2 classes (nodule or non-nodule).

3.3 Improvement on prior-net based pathology segmentation

3.3.1 Background

Segmentation of lung pathology in medical scans or images is vital for pulmonary disease diagnosis. However, the presence of different types of lung pathology with varying sizes,

shapes, and textures, as well as the visual similarity with the surrounding non-pathological tissues, make automatic lesion segmentation a challenging task to perform. Past effort focuses on using energy minimization methods [6, 14, 19] to segment out pathology, followed by using classifiers to reduce false positively detected regions [41, 20, 40]; as well as CNN-based techniques [8], encoder-decoder, or U-net architecture. Despite those model succeeding at performing segmentation, most of them still need to address the problems of intra-class inhomogeneity and boundary ambiguity of target regions. [2]

Joskowicz et al. [13] proposes the idea of segmentation prior, which encodes a particular type of prior knowledge such as shape, size, and appearance of the pathological region. This strategy has shown to improve the segmentation accuracy. Astaraki et al. [2] further proposes the idea of obtaining prior information by first capturing the prior knowledge for healthy lung structure instead of lesions, and further derive prior knowledge of pathology indirectly. Astaraki et al. [2] implements this idea by using 3 modules: the first module learns to reconstruct healthy lung CT scans via image inpainting with the pathology mask provided; the second module (a VAE) learns the distribution of healthy images in order to restore the pathological region to the healthy version of the lung scan; and, finally, the third module (Prior UNet) uses the differences between the original slice and the healthy version generated by module 2 as the detected anomaly, which is fed to a plain U-net as the prior knowledge to improve segmentation. Figure 3.5 shows the architecture of this proposed segmentaion method.

On the surface level, module 1 and 2 might seem to achieve the same thing, but the purpose of module 1 is to manually generate supervised learning samples for module 2. Module 2 then learns the underlying latent representations or prior information of healthy scans, which is then used to indirectly derive prior of pathology as proposed.

3.3.2 Implementation of Module 1 via RePaint

Following the same structure as Astaraki et al. [2], we propose to implement module 1, the in-painting model, using diffusion model instead of an autoencoder. We first train an Improved Denoising Diffusion Probabilistic Model [22] using all 32,040 healthy scans we have generated, as mentioned in Section 3.2.1. Unlike the PCNN shown in Astaraki et al. [2] or normal in-painting task, the diffusion model does not need random destructive mask as input during the learning process. Instead, we directly use RePaint [21], which uses pre-trained DDPM models to perform in-painting: giving the pathological image containing the nodule and the nodule mask, reconstruct the image such that it fills the pathology with healthy tissues. For the Improved Denoising Diffusion Probabilistic Model, we trained for 107,310 steps with the default configuration (linear schedule with diffusion steps of 1000) provided in the repository of Lugmayr et al. [21], with 4 samples per step. Figure 4.3 shows example images generated by the RePaint model.

3.3.3 Implemented Approaches of Module 2

The construction and training of module 2 have not been successful; nevertheless, several methods and models have been tried. Below is details descriptions of each experiment. The results and evaluations can be found in Section 4.2.3 of Chapter 4: Evaluation.

We have tried to replicate the effort in Astaraki et al. [2], which is to implement a Normal Appearance Autoencoder (NAA) model and train using pairs of pathological and healthy version of the same scan. The proposed NAA model is, essentially, a convolu-

tional Variational Autoencoder (VAE) with a special loss function: instead of only using a reconstruction loss and Kullback–Leibler divergence loss, the paper adds a new regularization term which is computed using the difference between the latent space encoding of the model input and the ground truth. Equation 3.1 shows the loss function used during training of the the NAA model. $\odot$ represents element-wise multiplication. Additionally, a weight mapping ω has been used which gives more weights within the lung region and less weights outside the lung region; α balances the effect of latent space regularization. In our setting, the input image size is 256 x 256, thus our NAA is one layer less than the original proposed version in Astaraki et al. 2022. See Appendix A.1 for details of the NAA architecture. Different from a normal VAE with linear resampling layers, a convolutional VAE uses Conv2d layers for the resampling of μ and σ. This will preserve the spatial correlation among pixels in the input image, thus will produce model predictions that correspond to the original image layout. In the end, this approach has failed to reconstruct high-fidelity healthy version of the pathological input, only being able to produce blurry and vague representation of the shape of the lung region. See Section 4.2.3 for the reconstructed images.

$$\mathcal{L} = \omega \odot \mathcal{L}_{rec} + \alpha \cdot \mathcal{L}_{reg} + \mathcal{L}_{prior} \tag{3.1}$$

$$= \omega \odot MSE(y, y\prime) + \alpha MSE(z(x), z(y)) + D_{KL}(z, N(0, I)) \tag{3.2}$$

For the purpose of experimentation, a normal autoencoder is also trained instead of using a variational autoencoder (VAE). For the bottleneck layer of the autoencoder, both linear and convolutional layers have been attempted, and the result is still blurry and vague, only containing the noisy overall structure of the lung region.

The second approach of implementing module 2 involves in using the Diffusion Autoencoder (Diffae) [29] to try to sample healthy images of the pathological input. The Diffae trains a diffusion probabilistic model (DPM) and tries to perform representation learning, thus also enabling a decoder to reconstruct or manipulate the input image. The goal is to capture the latent variable that represents the nodules and eliminates them in the output. There are 2 attempts to train a Diffae model: the first attempt is to train with paired image of pathological and healthy version of the same image, and the second attempt is to train only with healthy/nodule-free images. In the first attempt, the loss function is modified such that it's computed as the difference between the pathological input and the healthy expectation. However, the diffusion model is set up in a way such that the training loss is the difference between the Gaussian noise added on the reversely diffused image and the reversely diffused image itself; hence, it's unable to capture the difference. As a result, the diffusion autoencoder is good at reconstructing the input image, but is also still able to reconstruct the nodule, which is not desirable. The second attempt only trains Diffae on nodule-free images, with the loss function being the original diffusion model training loss. However, as expected, the Diffae is strong enough to produce high-fidelity output, including the nodules. For both attempts, the Diffae is trained over 200,000 steps, with learning rate being 0.0001, batch size being 2 due to GPU memory constraint, and a linear diffusion schedule of 1,000 steps. Examples of visual results are available in Section 4.2.3.

The third approach is to use Semantic Diffusion Model (SDM), but instead of using the semantic masking of the image, the class label is now the inpainted image generated by the RePaint model. Therefore, the diffusion process is trained based on the healthy version of the pathological image. The goal is to input a nodule slice and output a healthy version

```
Inside image_datasets.py:
...
elif dataset_mode == 'module2':
    images = sorted(glob(os.path.join(data_dir, f'inpainted/*.png')))
    labels = sorted(glob(os.path.join(data_dir, f'gt/*.png')))

    train_images = images[:1000]; train_labels = labels[:1000]
    val_images = images[1000:]; val_labels = labels[1000:]

    all_files = train_images if is_train else val_images
    classes = train_labels if is_train else val_labels
    instances = None
...

Inside train_util.py:
...
# The preprocess_input() function is re-written as below
def preprocess_input(self, data):
    data['label'] = data['label'].long()
    cond = {key: value for key, value in data.items() if key not in ['label',
    ↪  'instance', 'path', 'label_ori']}
    cond['y'] = data['label'] / 255

    return cond
...
```

Listing 4: Modifications to the SDM repository that implements the third approach for module 2.

of the same image. Listing 4 shows the necessary modifications made in the original SDM repository in order to implement the third approach. As a result, the third approach has also failed to produce nodule-free images, with results shown in Section 4.2.3.

Although following the instructions from Astaraki et al. [2], the model is not able to produce the same results shown in the paper. The underlying reason of VAE not being able to generate high-fidelity images will be investigated in the future.

3.3.4 Implementation of Module 3

Astaraki et al. [2] also provides an overall description of the architecture for the segmentation Prior-Net: a U-Net-like architecture with two-channel inputs in which the first channel represents the original image and the second channel is the corresponding prior image. The author assume that integrating the original image and prior one into a two-channel image would encourage the model's attention to predict more meaningful segmentation masks. The U-net includes an encoder of 4 conv blocks and a decoder of 4 transposed conv blocks, with each conv block containing a Conv2D layer with a kernel size of 3 and ReLu activation function, and a batch normalization layer followed by another Conv2D, batch normalization, and a dropout layer with the rate of 0.2. This conv block is the same as proposed in module 2 architecture. Listing 9 shows an example implementation/architecture of the architecture described in the paper.

```
--------------------------------------------------------------------------------
        Layer (type)               Output Shape      Param Count      Component
================================================================================
                                                                   *************
          Conv Block          [-1, 8, 256, 256]              152          ^
           MaxPool2d          [-1, 8, 128, 128]                0          |
          Conv Block         [-1, 16, 128, 128]            1,168          |
           MaxPool2d           [-1, 16, 64, 64]                0       Encoder
          Conv Block           [-1, 32, 64, 64]            4,640          |
           MaxPool2d           [-1, 32, 32, 32]                0          |
          Conv Block           [-1, 64, 32, 32]           18,496          |
           MaxPool2d           [-1, 64, 16, 16]                0          v
                                                                   *************
          Conv Block          [-1, 128, 16, 16]           73,856     Latent Space
                                                                   *************
          Conv Block           [-1, 64, 16, 16]           73,792          ^
     ConvTranspose2d           [-1, 64, 32, 32]           16,448          |
          Conv Block           [-1, 32, 32, 32]           18,464          |
     ConvTranspose2d           [-1, 32, 64, 64]            4,128          |
          Conv Block           [-1, 16, 64, 64]            4,624       Decoder
     ConvTranspose2d          [-1, 16, 128, 128]            1,040          |
          Conv Block           [-1, 8, 128, 128]            1,160          |
     ConvTranspose2d           [-1, 8, 256, 256]              264          v
                                                                   *************
              Conv2d           [-1, 1, 256, 256]                9      Output Layer
================================================================================
```

Listing 5: Model architecture information produced by torchsummary. The PyTorch implementation of this network is available in the repository of this project. Each Conv Block consists of a Conv2D layer with a kernel size of 3 and ReLu activation function, and a batch normalization layer followed by another Conv2D, batch normalization, and a dropout layer with the rate of 0.2.

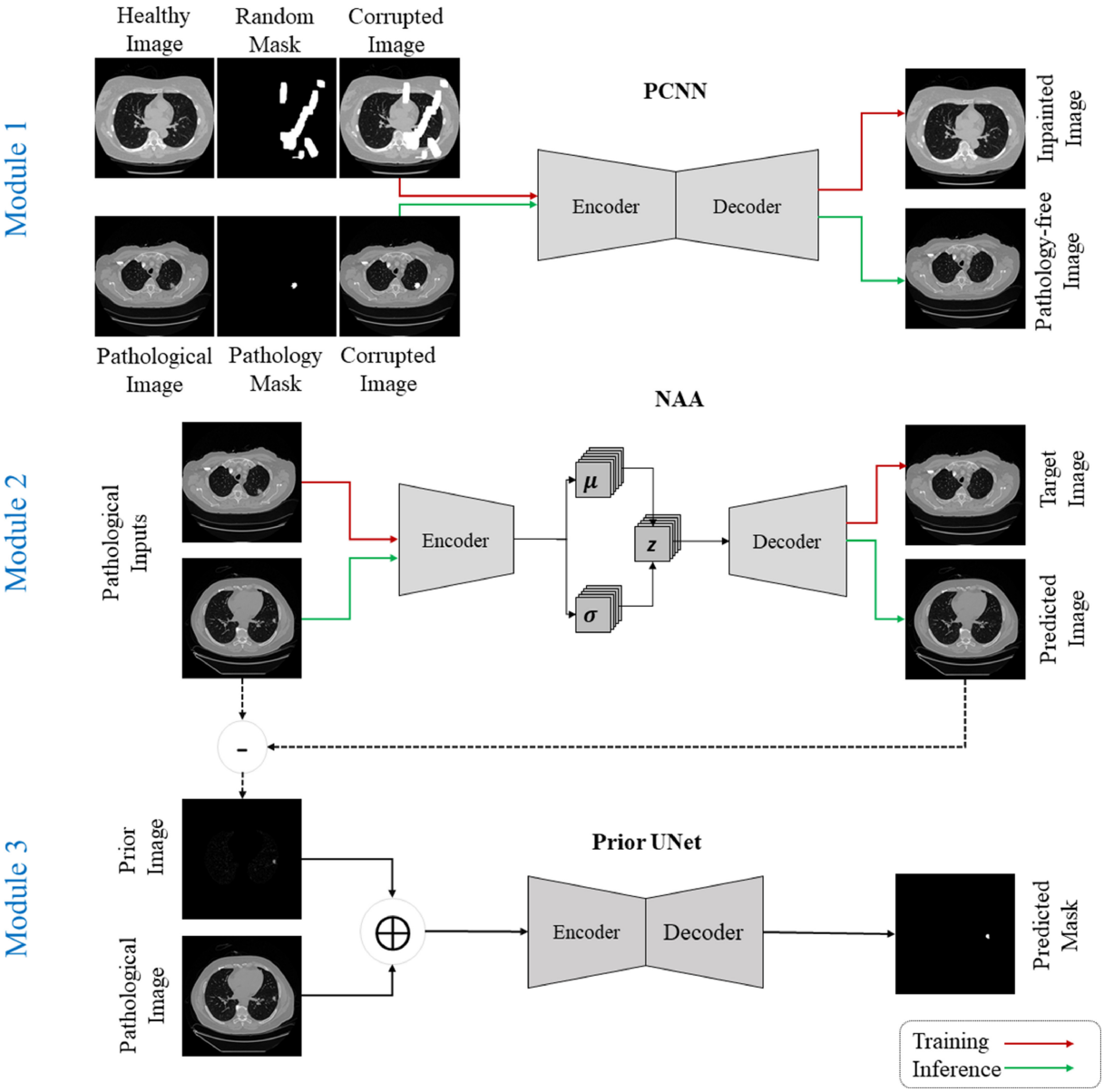

Figure 3.5: The architecture of the segmentation method proposed in Astaraki et al. [2]. Originally, module 1 is an in-painting model which can reconstruct the pathological region into healthy tissue; module 2 is a VAE which learns the distribution differences between pathological and healthy veresion of the same scan, and thus produces a prior image which is the difference between the input and the output; module 3 is the prior-net that performs segmentation based on the prior image.

Chapter 4

Evaluation

4.1 Experiments on Dataset Padding

4.1.1 Dataset and Configuration

The data preprocessing method in Section 3.2.1 results in 1139 slices containing nodules, and 32,040 nodule-free slices. For all experiments, the training and testing sets are divided by patient ID to ensure no data cross-contamination. Specifically, 744 patients were used for training, while 144 patients were used for testing. As nodule-free slices would greatly outnumber slices with nodules (almost 1 in 100), only 1 in 4 (empirically selected) nodule-free slices are selected to train the generative models, as well as subsequent downstream tasks. SDM and SPADE [23], a previous state-of-the-art method used for comparison, are trained and both are used to generate synthetic 2D pulmonary CT slices; 1000 slices with nodules, and another 1000 that are nodule-free.

Two downstream tasks, namely nodule detection and localization, are then trained with a mixture of synthetic and real samples. As mentioned in Section 3.2.1, SDM was trained with an image size of 256×256 (reduced from 512×512 due to GPU memory constraint when using SDM) and a batch size of 2. Training of SDM took approximately 2 days for 100,000 steps, using the AdamW optimizer with an initial learning rate of 0.0001; and inference speed of SDM is about 2 images per 15 minutes. SPADE was trained with an image resolution of 512×512 and a batch size of 16. Training of SPADE took approximately 7 hours for 20 epochs, using the Adam optimizer with an initial learning rate of 0.0001 as well. All experiments in this subsection were conducted on a machine with an Nvidia A6000 GPU. Later, it has been found that with an Nvidia GeForce RTX 4090 GPU, the training and inference speed can be improved by about 2-3 times.

4.1.2 Performance

All experiments are run for 10-folds with the synthetic images in the test fold being excluded if there are any, and significance of accuracy/AP50 is confirmed by Wilcoxon rank-sum test as shown in the p-value column. Table 4.1 shows the relevant metrics of two models before and after adding diffusion-generated images in the training set. Figure 3.2 shows samples generated by SDM and SPADE, and Figure 4.1 shows 2 example downstream nodule localizations.

For the nodule detection task, Table 1 shows that adding SDM-generated images increases both the accuracy and F1 score, as well as lower the standard deviation, when com-

	Accuracy (%)	Precision (%)	Rec./Sen. (%)	Specificity (%)	F1	p-value
I,A	85.76 ± 1.69	$\mathbf{85.63 \pm 3.66}$	86.42 ± 5.05	85.03 ± 6.10	85.83 ± 1.61	-
I,B	88.99 ± 1.32	83.3 ± 2.74	$\mathbf{90.64 \pm 2.86}$	87.86 ± 2.98	86.76 ± 1.31	82.0×10^{-6}
I,C	$\mathbf{89.72 \pm 1.26}$	85.09 ± 2.21	90.37 ± 2.14	$\mathbf{89.29 \pm 1.93}$	$\mathbf{87.61 \pm 1.23}$	$\mathbf{9.54 \times 10^{-6}}$
	AP_{50} (%)	AP_{60} (%)	AR_{50} (%)	AR_{60} (%)	AR_{70} (%)	p-value
II,A	80.26 ± 5.62	73.73 ± 6.03	89.23 ± 3.85	83.62 ± 4.12	64.96 ± 4.39	-
II,B	80.04 ± 4.60	72.37 ± 5.14	90.18 ± 3.52	83.52 ± 4.10	66.24 ± 3.50	0.985
II,C	$\mathbf{88.75 \pm 3.21}$	$\mathbf{84.80 \pm 3.72}$	$\mathbf{95.02 \pm 2.15}$	$\mathbf{91.55 \pm 2.49}$	$\mathbf{78.08 \pm 2.96}$	$\mathbf{4.825 \times 10^{-4}}$
	Accuracy (%)	Precision (%)	Rec./Sen. (%)	Specificity (%)	F1	p-value
III.-,A	82.67 ± 1.12	83.46 ± 1.55	84.32 ± 4.08	87.11 ± 3.40	84.01 ± 1.09	-
III.-,C	$\mathbf{86.47 \pm 1.67}$	$\mathbf{86.09 \pm 2.01}$	$\mathbf{87.12 \pm 2.43}$	$\mathbf{88.76 \pm 1.97}$	$\mathbf{86.31 \pm 1.32}$	7.65×10^{-6}
III.+,A	84.46 ± 1.22	85.16 ± 2.20	85.98 ± 3.09	89.23 ± 3.76	85.14 ± 1.09	-
III.+,C	$\mathbf{88.65 \pm 1.92}$	$\mathbf{87.52 \pm 2.11}$	$\mathbf{90.28 \pm 2.22}$	$\mathbf{91.02 \pm 2.01}$	$\mathbf{88.77 \pm 1.34}$	8.04×10^{-6}

Table 4.1: Relevant metrics on 4 experiments: I: Nodule detection task. II: Nodule localization task. III: nodule classification task, where - represents benign and + for malignant. A: Without synthetic images in train set. B: With SPADE images in train set. C: With diffusion images in train set. p-value is generated between A (control experiment) and other experiments (B or C), and is for accuracy in the nodule detection task while for AP_{50} in the nodule localization task. AP and AR are Average Precision and Recall, and the subscript denotes the IoU% used.

pared to the baseline and SPADE. For the nodule localization task, the model trained with the additional SDM-generated images outperformed both the baseline and the SPADE-image-trained model in AP and AR in all selected IoU test points. Achieving improvements of 3.96% for detection accuracy and 8.50% for AP_{50} in nodule localization task, respectively, demonstrates the feasibility of the approach. Finally, the FID scores for SDM among nodule and non-nodule diffusion-generated images are 80.820 and 84.494, respectively, and the FID scores of those for SPADE-generated images are 186.609 and 147.451. The FID scores show that SDM-generated images have significantly better quality than those generated by the previous SPADE method.

For the SDM conditioned on the malignancy score, the same classification model has been used as in the first downstream task, differing only in 3 classes (non-nodule, benign, and malignant) instead of 2 classes (non-nodule/nodule). Table 4.1 shows the relevant metrics of the classification task, with accuracy improved by 4.01%.

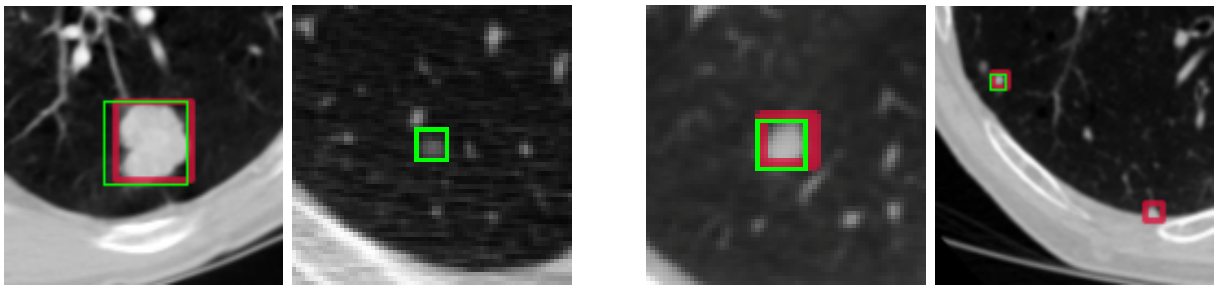

Figure 4.1: Localization downstream task. Left group: SDM, Right group: SPADE. In each group, the image on the left shows the correctly identified nodules, the image on the right shows false negative/positive detection. Green box is ground truth and red box is prediction.

4.1.3 Discussion

The FID score of SDM-generated images is much lower than SPADE-generated images, indicating the quality of synthetic images via SDM is significantly better than synthetic images via SPADE. However, this comes at a trade-off where generating synthetic images using SDM is much more time-consuming and computationally expensive compared to SPADE (i.e. $\sim$7 min/image for SDM whilst 320 images/min for SPADE using an Nvidia A6000 GPU). Surprisingly, in the SDM images, fine details in the trachea region of original images are preserved, while in the SPADE images the area is filled with random noises/strokes. Overall, our experiments have shown that SDM has the potential of generating high-fidelity pulmonary CT images, even with nodules of small diameters, as evident by the improvement of downstream tasks compared to SPADE and baseline.

Possibilities of running shorter steps or using the cosine schedule when inferencing from SDM using a DDPM model trained with linear schedule of 1000 diffusion steps have been experimented as well. Figure 4.2 shows images generated by using 200 steps or 1,000 full steps, with either linear or cosine schedules. Images generated using a cosine schedule is also inferenced on a model trained using a cosine schedule. In general, during inference, if the step count is lower than a certain threshold, the image quality will be reduced to certain degree. In this case, the default 1,000 diffusion steps serves as a good reference point, lower than which the image quality might degrade. The image generated by the cosine schedule appears gray and lacks important details such as the trachea, but the overall structure is contact.

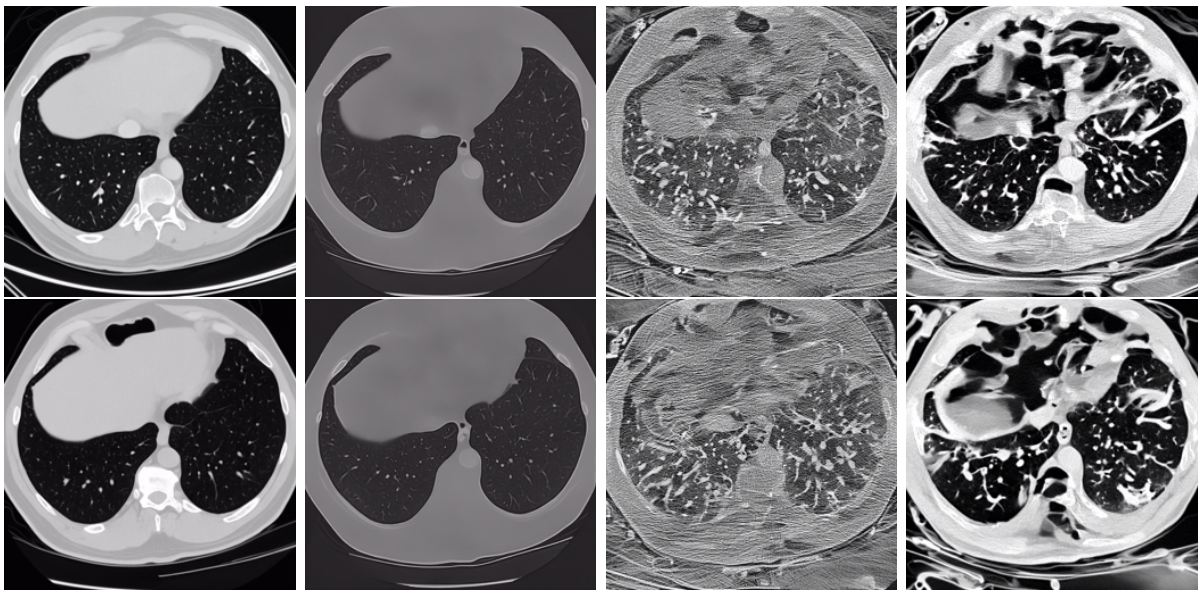

Figure 4.2: Two examples, each in a row, of generating using different steps and schedules. From left to right: image inferenced with linear schedule and 1000 steps (high-fidelity image as a reference), image inferenced with cosine schedule and 1000 steps, image inferenced with linear schedule and 200 steps, image inferenced with cosine schedule and 200 steps.

```
q05 = np.quantile(lesion_free_image, 0.05)
q95 = np.quantile(lesion_free_image, 0.95)
lesion_free_image = np.clip(lesion_free_image, q05, q95)

q05 = np.quantile(lesion_image, 0.05)
q95 = np.quantile(lesion_image, 0.95)
lesion_image = np.clip(lesion_image, q05, q95)
```

Listing 6: Modified version of the code in Listing 1. total_mask is the final segmentation mask. average_malignancy.csv is a file prepared by the script in Appendix A.3. The final mask is stored as a png image whose visual appearance is all black since all pixel values are only within range 0 to 6 out of 255.

4.2 Experiments on Prior-Net Based Pathology Segmentation

4.2.1 Dataset and Configuration

We used LUNA16 dataset to produce some results generated by module 1, the RePaint model, and module 2, which involves two unsuccessful attempts of recovering the nodules in the image.

4.2.2 Performance of Module 1: RePaint

Figure 4.3 shows two examples generated by the RePaint model. The model accepts two input images: one is the original scan with a nodule, and the other is a segmentation mask whose pixels equal to 255 in known region and 0 in the region that needs to be in-painted. As seen from the rightmost column, the in-painted nodules resemble the appearance of normal lung tissues, but a discrete white border around the nodule region is also formed which produce visual discontinuity among the surrounding tissues. This might be mitigated by expanding the nodule masks such that the RePaint model is able to consider surrounding pixels during in-painting. Another subtle detail of great importance is that in the in-painted image, the known region is visually the same as the original input; however, after visualizing the difference between the two image, it shows that some pixels have different intensity values from the original image. Therefore, when handling the in-painted image, a thresholding technique must be applied such that pixels below $Q_{p=0.05}$ and above $Q_{p=0.95}$ are clipped. After this processing step, the image difference will only contain the nodule. Listing 6 shows the code to perform this clipping step, and Figure 4.4 shows the image differences before and after applying this clipping step.

4.2.3 Performance of Module 2 via 3 Implemented Approaches

For module 2, three approaches have been tried, and all three approaches have reconstructed the nodule due to model being over-capable. Figure 4.5 shows the results of the first approach using Normal Appearance Autoencoder (NAA), Figure 4.6 shows the results of the second approach using the Diffusion Autoencoder (Diffae), and Figure 4.7 shows the results of using SDM without semantic masks. Both the NAA and the Diffae model cannot learn the latent variable representing the nodules and thus remove them.

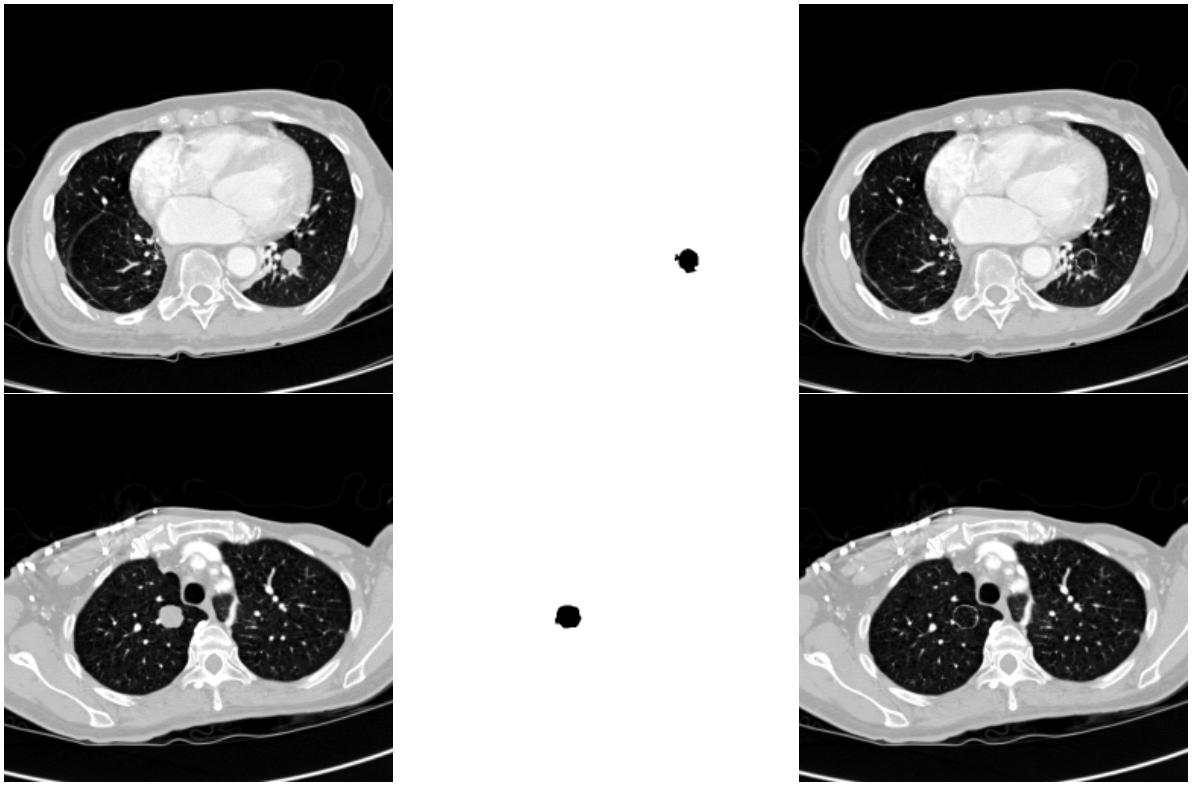

Figure 4.3: Example original and in-painted images by the RePaint model. L-to-R: Original CT image, mask (black) only for the nodule region, and image after in-painting with healthy tissue.

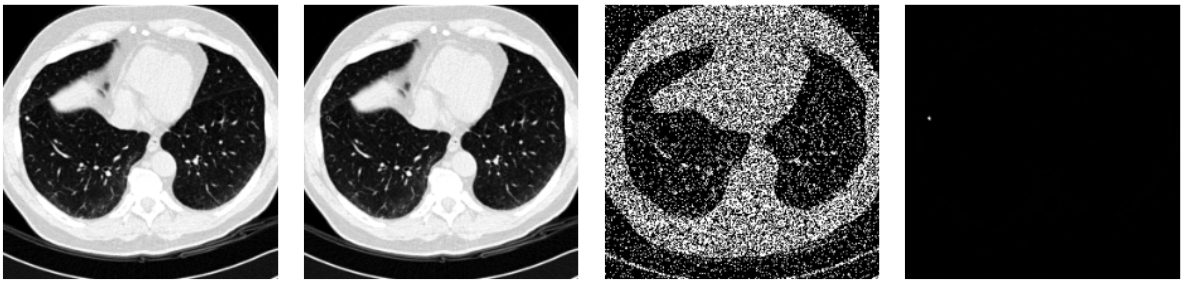

Figure 4.4: Example of using the clipping process mentioned above. From left to right: 1. The original pathological image containing a small nodule near the border of the left lung. 2. The nodule-free version of the original image. 3. The image difference between image 1 and 2. 4. The image difference between image 1 and 2 after the two images are processed by clipping. After clipping, the noises in image 3 have been removed and the difference of two images will only contain the nodule.

For the third approach, it has been noticed that the model does not always construct the nodule at the original location: it sometimes sporadically adds a nodule in another location. Overall, all three approaches failed to produce expected result, which is nodule-free version of the pathological image. However, due to the limited time span of this project and the variational nature of the VAE model, only speculated reasons are provided and discussed. Further investigation needs to be done in order to provide comprehensive

explanation behind the results shown in the figures.

For the Diffae approach, the image quality has been evaluated as the FID score being 197.461 after training for 200,000 samples, suggesting a reasonably good image quality.

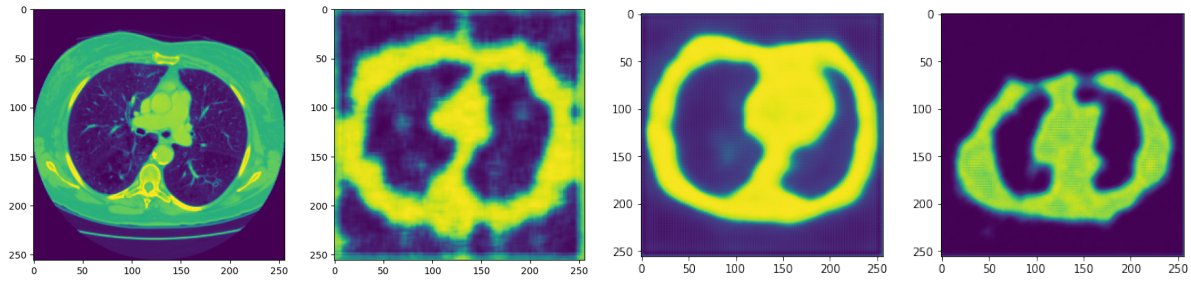

Figure 4.5: Results generated by the Nornal Appearance Autoencoder (NAA) model. From left to right: first two images are a pair of the original slice and the result produced by the NAA model, followed by another two results produced by the NAA model.

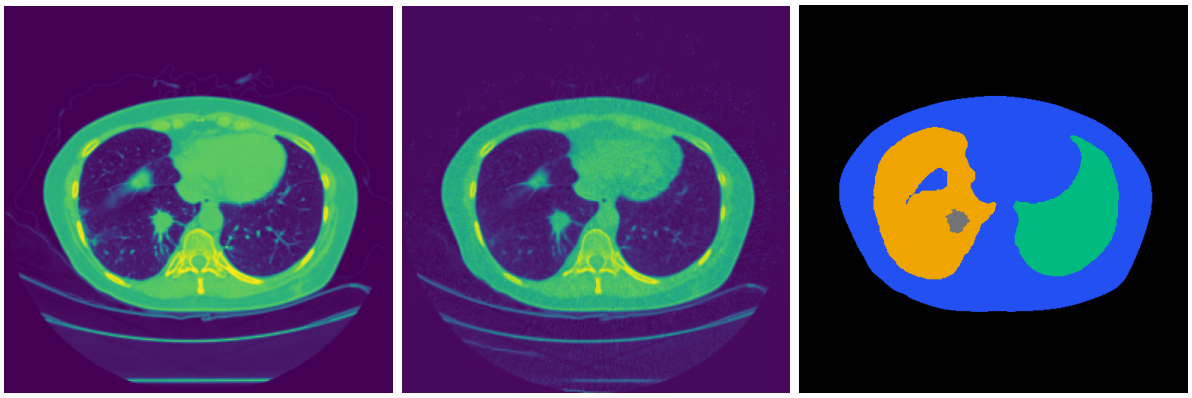

Figure 4.6: Results generated by the Diffusion Autoencoder (Diffae) model. Left: the original pathological slice. Middle: result generated by the Diffusion Autoencoder (Diffae). Right: mask showing different parts of the lung. The colour scheme of the mask is the same as in Figure 3.2

4.2.4 Comments on Module 3

Module 3 is supposed to be the Prior-Net that performs segmentation of pathology. It has not been evaluated due to module 2 being failed to produce expected results. If implemented, it should be evaluated by the Intersection-over-Union (IoU) and Dice Coefficient (F1 Score).

4.3 Efficiency

Currently, diffusion model runs slower compared to other well-known models such as GAN or VAE. Relevant training speed has been mentioned in Section 4.1.1. For inferences using an Nvidia A6000 GPU, the speed of SDM with the image size of 256 x 256 is about 7 min/image while the speed of SPADE is about 320 images/min. Using the latest Nvidia

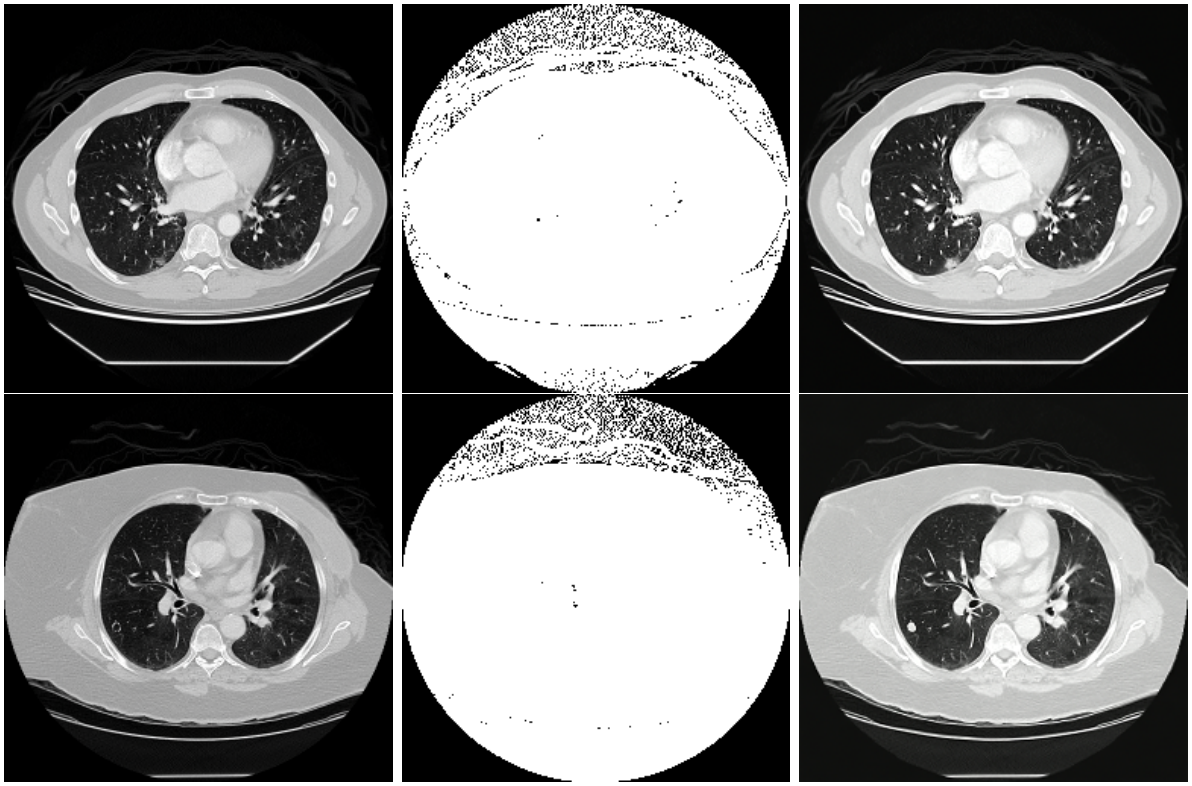

Figure 4.7: Results generated by the SDM trained using healthy images without semantic masks (third approach).

GTX 4090 GPU (single), the inference speed using SDM with the same image size is about 3.5 min/image, whereas SPADE generates over 400 images/min. The memory required to train and inference via diffusion model is also huge, around 23.3 GB with batch size set to 2. Using DDIM [34], it is estimated that the training and inference speed will be accelerated up to 50 times.

4.4 Concerns and Potential Problems

Although images generated via diffusion model produce promising results on downstream task performances, it still has some limitations and constraints. First, the medical semantic behind the synthetic samples must be confirmed with expert radiologists or medical experts. Although being already significantly improved from GAN, in some images, some tissues might not be correctly generated or missing. Second, the generation of the image heavily relies on the underlying dataset and maskings being used. In our experiments, we only used LUNA16 dataset to generate, which might be biased and cannot capture the true distribution patients' scans. The choice of masking is also vital: we have chosen five types of masking for the lung region, which works fine; however, for CT scans of other body parts, it might need more fine-grained masking, namely including more components or types of masks which might not always be available or easy to generate.

Another potential problem is the labelling of machine-generated scans. Data has to be carefully processed so that the real samples and synthetic samples are not mistaken: with the improvement via diffusion model, the synthetic samples look visually realistic

and even contain plausible human body structures, but it should not be directly treated as a valid human scan.

Chapter 5

Conclusion

5.1 Project Summary

In this project, we have accomplished two major tasks:

- A Semantic Diffusion Model capable of producing high-fidelity pulmonary CT scans guided by the segmentation mask/semantic map of the input image. The high-definition images can be used to perform dataset padding such that downstream model performances can be greatly improved. Diffusion model trained and inferenced with various schedule and steps are also experimented.

- An incomplete attempt of replicating the effort by Astaraki et al. [2], but instead using diffusion model. We have successfully re-implemented Module 1 using the RePaint [21] model, but have been unable to implement Module 2 albeit having tried 3 different approaches to tackle the issue of generating blurry/vague images: constructing the original Normal Appearance Autoencoder (NAA) model, using the Diffusion Autoencoder (Diffae), and using the SDM without giving the semantic mask.

Both tasks are evaluated using suitable metrics:

- For the first task of dataset padding, our experiments have shown that SDM has the potential of generating high-fidelity pulmonary CT images, even with nodules of small diameters, as evident by the improvement of downstream tasks compared to SPADE and baseline. We use FID score, accuracy, precision, recall, specificity, and F1 score, as well as the p-value produced by the Wilcoxon rank-sum test for measuring the nodule detection downstream task improvement, and use Average Precision/Recall and also the Wilcoxon rank-sum test for measuring the nodule localization downtream task improvement. Additionally, we have used malignancy information from LIDC dataset to train another SDM conditioned on the malignancy score smaller or bigger than a threshold; the conditional SDM generates benign/-malignant images that are also evaluated using the same set of metrics under the classification task.

- For the extension of implementing a Prior-Net based segmentation task, more investigations are needed to resolve the failure of constructing and successfully training module 2. The FID score for Diffae approach has been provided.

5.2 Publication

The first task of this project has been published on Medical Imaging with Deep Learning (MIDL) 2023 conference as a short paper under the title *High-Fidelity Image Synthesis from Pulmonary Nodule Lesion Maps using Semantic Diffusion Model*. The author of this project will give an virtual poster presentation of relevant findings in this project.

5.3 Future Work

For the first task of generating high-fidelity samples, relevant future work includes training the SDM in 2.5D in order to perform 3D volume generation, training another SDM conditioned on information more than just the malignancy score, or even perform longitudinal study to generate nodules of different sizes, shapes, and phases.

For the second task of performing prior-aware pathology segmentation task, investigations towards latent space variable capture as well as model capacity are needed, and future evaluation steps need to be performed as well, as suggested in Section 4.2.4.

Appendix A

Appendix

A.1 Normal Appearance Autoencoder Model Architecture

Block Name	Size	# Filters	Detail
input	512 × 512	1	-
1st conv.	256 × 256	8	Conv-block(k=3, relu), max-pooling(2), dropout(0.2)
2nd conv.	128 × 128	16	Conv-block(k=3, relu), max-pooling(2), dropout(0.2)
3rd conv.	64 × 64	32	Conv-block(k=3, relu), max-pooling(2), dropout(0.2)
4th conv.	64 × 64	64	Conv-block(k=3, relu), dropout(0.2)
5th conv.	64 × 64	128	Conv-block(k=3, relu), dropout(0.2)
Sampling layer	64 × 64	128	2 Convd2D(k=3, relu) for mean and log-var layers
1st conv. trans	128 × 128	128	Conv2DTrans(k=3, s=2, relu), dropout(0.2), Conv-block(k=3, relu)
2nd conv. trans	256 × 256	64	Conv2DTrans(k=3, s=2, relu), dropout(0.2), Conv-block(k=3, relu)
3rd conv. trans	512 × 512	32	Conv2DTrans(k=3, s=2, relu), dropout(0.2), Conv-block(k=3, relu)
4th conv. trans	512 × 512	16	Conv2DTrans(k=3, s=1, relu), dropout(0.2), Conv-block(k=3, relu)
5th conv. trans	512 × 512	8	Conv2DTrans(k=3, s=1, relu), dropout(0.2), Conv-block(k=3, relu)
output	512 × 512	1	Convd2D(k=1, sigmoid)

Table A.1: Scheme of the NAA model architecture. In this table, each Conv-block consists of a Convd2D layer, a batch normalization layer, another conv2D layer followed by another batch normalization, respectively. K refers to the kernel size, and s indicates the stride length.

A.2 Training and Inference Configuration of the Nodule Classification and Localization Tasks

A.2.1 Configuration of the Nodule Classification Task

```
_base_ = "seresnet/seresnet101_8xb32_in1k.py"

dataset_type = 'CustomDataset'
classes = ['nodule', 'non_nodule']  # The category names of your dataset

model = dict(head=dict(num_classes=2, topk=(1,)))

train_pipeline = [
    dict(type='LoadImageFromFile'),
```

```
    dict(type='Resize', size=(224, -1), backend='pillow'),
    dict(
        type='Normalize',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        to_rgb=True),
    dict(type='ImageToTensor', keys=['img']),
    dict(type='ToTensor', keys=['gt_label']),
    dict(type='Collect', keys=['img', 'gt_label'])
]

test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='Resize', size=(224, -1), backend='pillow'),
    dict(
        type='Normalize',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        to_rgb=True),
    dict(type='ImageToTensor', keys=['img']),
    dict(type='Collect', keys=['img'])
]

data = dict(
    train=dict(
        type=dataset_type,
        data_prefix='../LUNA16_new/train',
        ann_file=None,
        classes=classes
    ),
    val=dict(
        type=dataset_type,
        data_prefix='../LUNA16_new/test',
        ann_file=None,
        classes=classes
    ),
    test=dict(
        type=dataset_type,
        data_prefix='../LUNA16_new/test',
        ann_file=None,
        classes=classes
    )
)

default_hooks = dict(
    checkpoint = dict(type='CheckpointHook', interval=25)
)

evaluation = dict(interval=1, metric='accuracy', metric_options=dict(topk=(1,)))
```

Listing 7: Configuration of the model, pre-processing method, and performance evaluation method used in the nodule classification task.

A.2.2 Configuration of the Nodule Localization Task

```
# The new config inherits a base config to highlight the necessary modification
_base_ = 'faster_rcnn/faster_rcnn_x101_32x4d_fpn_1x_coco.py'
```

```
3
4  # We also need to change the num_classes in head to match the dataset's annotation
5  model = dict(
6      roi_head=dict(
7          bbox_head=dict(num_classes=1),
8      )
9  )
10
11 train_pipeline = [
12     dict(type='LoadImageFromFile'),
13     dict(type='LoadAnnotations', with_bbox=True),
14     dict(type='Resize', img_scale=(1024, 1024), keep_ratio=True),
15     dict(type='RandomFlip', flip_ratio=0.5),
16     dict(
17         type='Normalize',
18         mean=[123.675, 116.28, 103.53],
19         std=[58.395, 57.12, 57.375],
20         to_rgb=True),
21     dict(type='Pad', size_divisor=32),
22     dict(type='DefaultFormatBundle'),
23     dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels'])
24 ]
25
26 test_pipeline = [
27     dict(type='LoadImageFromFile'),
28     dict(
29         type='MultiScaleFlipAug',
30         img_scale=(1024, 1024),
31         flip=False,
32         transforms=[
33             dict(type='Resize', keep_ratio=True),
34             dict(type='RandomFlip'),
35             dict(
36                 type='Normalize',
37                 mean=[123.675, 116.28, 103.53],
38                 std=[58.395, 57.12, 57.375],
39                 to_rgb=True),
40             dict(type='Pad', size_divisor=32),
41             dict(type='ImageToTensor', keys=['img']),
42             dict(type='Collect', keys=['img'])
43         ])
44 ]
45
46 img_prefix = '../data/luna16_sliced'
47
48 # Modify dataset related settings
49 dataset_type = 'COCODataset'
50 classes = ('nodule',)
51 data = dict(
52     train=dict(
53         img_prefix=img_prefix,
54         classes=classes,
55         ann_file=img_prefix + '/train_kfold_diffusion_0.json',
56         pipeline=train_pipeline
57     ),
58     val=dict(
59         img_prefix=img_prefix,
60         classes=classes,
```

```
        ann_file=img_prefix + '/test_kfold_diffusion_0.json',
        pipeline=test_pipeline
    ),
    test=dict(
        img_prefix=img_prefix,
        classes=classes,
        ann_file=img_prefix + '/test_kfold_diffusion_0.json',
        pipeline=test_pipeline
    )
)

runner = dict(type='EpochBasedRunner', max_epochs=12)

evaluation = dict(interval=1, metric='bbox')
```

Listing 8: Configuration of the model, pre-processing method, and performance evaluation method used in the nodule localization task.

A.3 Code for Generating Average Malignancy Map

```
# This function is used to generate mapping between series_id and file_path and
↪  generate mapping.csv
def generate_seriesuid_to_file_mapping():
  xml_files = list_image_files_recursively(".")
  series_id_to_file_path = {}

  def find_series_id_in_file(file_path, series_id):
    # Parse the XML file and get the root element
    tree = ET.parse(file_path)
    root = tree.getroot()

    for i in root.iter():
      if i.tag == '{http://www.nih.gov}SeriesInstanceUid':
        if (i.text == series_id):
          series_id_to_file_path[series_id] = file_path

  with open('annotations.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for i, row in enumerate(csv_reader):
      for file in xml_files:
        find_series_id_in_file(file, row['seriesuid'])
    csv_file.close()

  # Generate the mapping between series_id and file_path
  with open("mapping.csv", "w") as f:
    for k, v in series_id_to_file_path.items():
      f.write("%s,%s \n" % (k, v))
    f.close()

# This function is used: For each entry in annotations.csv, find the corresponding
↪  file path
# using the mapping generated above, match the coordZ field with the
↪  imageZposition field
# in the xml file, and save the malignancy field that corresponds to that entry
# in the annotations.csv file.
def generate_malignancy_mapping():
```

```
  seriesuid_to_malignancy = {}

  with open("annotations.csv", "r") as f:
    csv_reader = csv.DictReader(f)
    for i, row in enumerate(csv_reader):
      seriesuid = row['seriesuid']

      # Find the z_coord for the seriesuid
      z_coord = row['coordZ']
      with open("mapping.csv", "r") as mapping_file:
        csv_reader_mapping = csv.DictReader(mapping_file)
        for r in csv_reader_mapping:
          if (r['seriesuid'] == seriesuid):
            file_path = r['file_path']
        mapping_file.close()

      with open(file_path, "r") as xml_file:
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for i in root.iter():
          if i.tag == '{http://www.nih.gov}unblindedReadNodule':
            zPos = i.findall("{http://www.nih.gov}roi")
            zPos = [z.findall("{http://www.nih.gov}imageZposition") for z in zPos]
            zPos = [item.text for sublist in zPos for item in sublist]

            if any([abs(float(z) - float(z_coord)) < 0.5 for z in zPos]):
              for j in i.iter():
                if j.tag == '{http://www.nih.gov}characteristics':
                  malignancy = j.iterfind('{http://www.nih.gov}malignancy')
                  malignancy = [m for m in malignancy]
                  if len(malignancy) > 0:
                    l = len(seriesuid_to_malignancy.items())
                    seriesuid_to_malignancy[l] = [seriesuid, malignancy[0].text,
                    ↪  xml_file.name, z_coord]
                    # break
        xml_file.close()
    f.close()

  with open("malignancy.csv", "w") as f:
    for k, v in seriesuid_to_malignancy.items():
      f.write("%s,%s,%s,%s,%s \n" % (k, v[0], v[1], v[2], v[3]))
    f.close()

def average_malignancy():
  with open("malignancy.csv", "r") as f:
    csv_reader = csv.DictReader(f)
    malignancy_map = {}
    for i, row in enumerate(csv_reader):
      if row['seriesuid'] in malignancy_map:
        if row['z_coord'] in malignancy_map[row['seriesuid']]:
          malignancy_map[row['seriesuid']][row['z_coord']]
                        .append(float(row['malignancy']))
        else:
          malignancy_map[row['seriesuid']][row['z_coord']] =
          ↪  [float(row['malignancy'])]
      else:
        malignancy_map[row['seriesuid']] = {row['z_coord']:
        ↪  [float(row['malignancy'])]}
```

```
    f.close()

  with open("average_malignancy.txt", "w") as f:
    for k, v in malignancy_map.items():
      for k1, v1 in v.items():
        f.write("%s,%s,%s\n" % (k, k1, sum(v1)/len(v1)))

# Call the functions in this order
generate_seriesuid_to_file_mapping() # This function is used to generate
↪  mapping.csv
generate_malignancy_mapping()        # This function is used to generate
↪  malignancy.csv
average_malignancy()                 # This function is used to generate
↪  average_malignancy.csv
```

Listing 9: Configuration of the model, pre-processing method, and performance evaluation method used in the nodule localization task.

List of Figures

Bibliography

[1] Samuel G. Armato III, Geoffrey McLennan, Luc Bidaut, Michael F. McNitt-Gray, Charles R. Meyer, Anthony P. Reeves, Binsheng Zhao, Denise R. Aberle, Claudia I. Henschke, Eric A. Hoffman, Ella A. Kazerooni, Heber MacMahon, Edwin J. R. van Beek, David Yankelevitz, Alberto M. Biancardi, Peyton H. Bland, Matthew S. Brown, Roger M. Engelmann, Gary E. Laderach, Daniel Max, Richard C. Pais, David P.-Y. Qing, Rachael Y. Roberts, Amanda R. Smith, Adam Starkey, Poonam Batra, Philip Caligiuri, Ali Farooqi, Gregory W. Gladish, C. Matilda Jude, Reginald F. Munden, Iva Petkovska, Leslie E. Quint, Lawrence H. Schwartz, Baskaran Sundaram, Lori E. Dodd, Charles Fenimore, David Gur, Nicholas Petrick, John Freymann, Justin Kirby, Brian Hughes, Alessi Vande Casteele, Sangeeta Gupte, Maha Sallam, Michael D. Heath, Michael H. Kuhn, Ekta Dharaiya, Richard Burns, David S. Fryd, Marcos Salganicoff, Vikram Anand, Uri Shreter, Stephen Vastagh, Barbara Y. Croft, and Laurence P. Clarke. The lung image database consortium (lidc) and image database resource initiative (idri): A completed reference database of lung nodules on ct scans. *Medical Physics*, 38(2):915–931, 2011. doi: https://doi.org/10.1118/1.3528204. URL https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3528204.

[2] Mehdi Astaraki, Örjan Smedby, and Chunliang Wang. Prior-aware autoencoders for lung pathology segmentation. *Medical Image Analysis*, 80:102491, 2022. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2022.102491. URL https://www.sciencedirect.com/science/article/pii/S1361841522001384.

[3] Maria JM Chuquicusma, Sarfaraz Hussein, Jeremy Burt, and Ulas Bagci. How to fool radiologists with generative adversarial networks? a visual turing test for lung cancer diagnosis. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 240–244. IEEE, 2018.

[4] MMPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark. https://github.com/open-mmlab/mmpretrain, 2023.

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[6] Aly Farag, Hossam Munim, and James Graham. A novel approach for lung nodules segmentation in chest ct using level sets. *Image Processing, IEEE Transactions on*, 22:5202–5213, 12 2013. doi: 10.1109/TIP.2013.2282899.

[7] Dachuan Gao, Xiaodan Ye, Xuewen Hou, Yang Chen, Xue Kong, Yuanzhong Xie, and Shengdong Nie. A method for distinguishing benign and malignant pulmonary nodules based on 3D dual path network aided by k-means clustering analysis. *J. Zhejiang Univ. Sci. B*, 23(11):957–967, November 2022.

[8] Sarah E Gerard, Jacob Herrmann, David W Kaczka, Guido Musch, Ana Fernandez-Bustamante, and Joseph M Reinhardt. Multi-resolution convolutional neural networks for fully automated segmentation of acutely injured lungs in multiple species. *Med. Image Anal.*, 60(101592):101592, February 2020.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[12] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[13] L. Joskowicz, D. Cohen, N. Caplan, and J. Sosna. Automatic segmentation variability estimation with segmentation priors. *Medical Image Analysis*, 50:54–64, 2018. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2018.08.006. URL https://www.sciencedirect.com/science/article/pii/S1361841518306546.

[14] Wei Ju, Dehui Xiang, Bin Zhang, Lirong Wang, Ivica Kopriva, and Xinjian Chen. Random walk and graph cut for co-segmentation of lung tumor on PET-CT images. *IEEE Trans. Image Process.*, 24(12):5854–5867, December 2015.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[16] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] Steve Kommrusch and Louis-Noël Pouchet. Synthetic lung nodule 3d image generation using autoencoders. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2019. doi: 10.1109/IJCNN.2019.8852224.

[19] Toshiro Kubota, Anna K Jerebko, Maneesh Dewan, Marcos Salganicoff, and Arun Krishnan. Segmentation of pulmonary nodules of various densities with morphological approaches and convexity models. *Med. Image Anal.*, 15(1):133–154, February 2011.

[20] Lin Lu, Yongqiang Tan, Lawrence H Schwartz, and Binsheng Zhao. Hybrid detection of lung nodules on CT scan images. *Med. Phys.*, 42(9):5042–5054, September 2015.

[21] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

[22] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[23] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[24] Aria Pezeshk, Berkman Sahiner, Rongping Zeng, Adam Wunderlich, Weijie Chen, and Nicholas Petrick. Seamless insertion of pulmonary nodules in chest ct images. *IEEE Transactions on Biomedical Engineering*, 62(12):2812–2827, 2015. doi: 10.1109/TBME.2015.2445054.

[25] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F Da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Ldm 100k dataset.

[26] Walter HL Pinaya, Mark S Graham, Robert Gray, Pedro F Da Costa, Petru-Daniel Tudosiu, Paul Wright, Yee H Mah, Andrew D MacKinnon, James T Teo, Rolf Jager, et al. Fast unsupervised brain anomaly detection and segmentation with diffusion models. *arXiv preprint arXiv:2206.03461*, 2022.

[27] Walter HL Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F Da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M Jorge Cardoso. Brain imaging generation with latent diffusion models. In *MICCAI Workshop on Deep Generative Models*, pages 117–126. Springer, 2022.

[28] Walter Hugo Lopez Pinaya, Petru-Daniel Tudosiu, Robert Gray, Geraint Rees, Parashkev Nachev, Sébastien Ourselin, and M Jorge Cardoso. Unsupervised brain anomaly detection and segmentation with transformers. *arXiv preprint arXiv:2102.11650*, 2021.

[29] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[30] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[32] Imran Shafi, Sadia Din, Asim Khan, Isabel De La Torre Díez, Ramón del Jesús Palí Casanova, Kilian Tutusaus Pifarre, and Imran Ashraf. An effective method for lung cancer diagnosis from ct scan using deep learning-based support vector network.

Cancers, 14(21), 2022. ISSN 2072-6694. doi: 10.3390/cancers14215457. URL https://www.mdpi.com/2072-6694/14/21/5457.

[33] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. Cancer statistics, 2017. *CA: A Cancer Journal for Clinicians*, 67(1):7–30, 2017. doi: https://doi.org/10.3322/caac.21387. URL https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21387.

[34] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[36] Weilun Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Dong Chen, Lu Yuan, and Houqiang Li. Semantic image synthesis via diffusion models. *arXiv preprint arXiv:2207.00050*, 2022.

[37] Lilian Weng. From gan to wgan. *lilianweng.github.io*, 2017. URL https://lilianweng.github.io/posts/2017-08-20-gan/.

[38] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018. URL https://lilianweng.github.io/posts/2018-08-12-vae/.

[39] Julia Wolleb, Florentin Bieder, Robin Sandkühler, and Philippe C Cattin. Diffusion models for medical anomaly detection. *arXiv preprint arXiv:2203.04306*, 2022.

[40] Dijia Wu, Le Lu, Jinbo Bi, Yoshihisa Shinagawa, Kim Boyer, Arun Krishnan, and Marcos Salganicoff. Stratified learning of local anatomical context for lung nodules in ct images. pages 2791–2798, 06 2010. doi: 10.1109/CVPR.2010.5540008.

[41] Ning Xiao, Yan Qiang, Muhammad Bilal Zia, Sanhu Wang, and Jianhong Lian. Ensemble classification for predicting the malignancy level of pulmonary nodules on chest computed tomography images. *Oncol. Lett.*, 20(1):401–408, July 2020.